4 Gaussian Pulses

4.1 Gaussian Pulse Sections

4.2 Gaussian Pulse Functions

4.3 Gaussian Pulse Figures & Tables

4.4 Gaussian Pulse Programs

4.5 Gaussian Pulse Propagators

4.5.1 Gaussian

Usage:

```
#include <P_Gaussian.h>
gen_op Gaussian(spin_system& sys, gen_op& H, String& Iso,
                                        double td, double theta, double phi=0.0)
void Acquire1D(gen_op &Op, gen_op &H, double dt=0)
void Acquire1D(gen_op &Op, super_op &L, double dt=0)
void Acquire1D(const Acquire1D &ACQ1)
```

Description:

The function *Acquire1D* is used to create a 1-dimensional acquisition computational core.

1. Acquire1D() - Creates an “empty” NULL acquire1D. Can be later filled by an assignment.
2. Acquire1D(gen_op &Op, gen_op &H, double dt) - Called with the operator for which the expectation values are desired (Op) and the static Hamiltonian (H) under which the system density operator will evolve, this function constructs an appropriate acquire1D for future computation use. The optional value dt may be set for an incremental delay time. This produces an exponential Liouvillian for rapid generation of time domain spectra.
3. Acquire1D(gen_op &Op, super_op &L, double dt) - Called with the operator for which the expectation values are desired (Op) and the system Liouvillian (L) under which the system density operator will evolve, this function constructs an appropriate acquire1D for future computation use. The optional value dt may be set for an incremental delay time. This produces an exponential Liouvillian for rapid generation of time domain spectra.
4. Acquire1D(const Acquire1D &ACQ1) - Called with another acquire1D quantity this function constructs an identical acquire1D to the input ACQ1.

Return Value:

Acquire1D returns no parameters. It is used strictly to create an acquire1D.

Examples:

See Also: =

4.5.2 =

Usage:

```
#include <P_Gaussian.h>
void acquire1D operator = (acquire1D &ACQ1)
```

Description:

The unary operator = (the assignment operator) allows for the setting of one acquire1D to another acquire1D. If the acquire1D being assigned to exists it will be overwritten by the assigned acquire1D.

Return Value:

None, the function is void

Examples:

See Also: acquire1D

4.5.3 Gpulse_Hs

Usage:

```
#include <P_Gaussian.h>
void Gpulse_Hs(gen_op* Hs, gen_op& Ho, gen_op& Fxy,
                                        int N, double ang, double tp, double fact)
```

Description:

The function ***Gpulse_Hs*** generates a series of active Hamiltonians applicable to a Gaussian shaped pulse. The Hamiltonians are defined only in the rotating frame of the rf-field of the pulse. The array of general operators ***Hs*** is filled with ***N*** operators representing the Gaussian waveform. The waveform consists of ***N*** steps, is of length ***tp*** (sec.), begins at intensity ***fact*** (decimal% of maximum), and will rotate magnetization on resonance by angle ***ang*** (degrees). The operator ***Fxy*** sets the pulse phase and selectivity. The operator ***Ho*** is the isotropic system Hamiltonian.

Return Value:

The function is void, it will fill operator array Hs.

Example:

```
#include <P_Gaussian.h>
Acquire1D ACQ1(det,L,sigmaeq);                // Set up for 1D acquisition
```

See Also:

4.5.4 Gpulse_Us

Usage:

```
#include <P_Gaussian.h>
void Gpulse_Us(gen_op* Us, gen_op& Ho, gen_op& Fxy,
                                        int N, double ang, double tp, double fact)
```

Description:

The function ***Gpulse_Us*** generates a series of propagators applicable to a Gaussian shaped pulse. The propagators are defined only in the rotating frame of the rf-field of the pulse. The array of general operators ***Us*** is filled with ***N*** propagators representing the Gaussian waveform. The waveform consists of ***N*** steps, is of length ***tp*** (sec.), begins at intensity ***fact*** (decimal% of maximum), and will rotate magnetization on resonance by angle ***ang*** (degrees). The operator ***Fxy*** sets the pulse phase and selectivity. The operator ***Ho*** is the isotropic system Hamiltonian.

Return Value:

The function is void, it will fill operator array Hs.

Example:

```
#include <P_Gaussian.h>
Acquire1D ACQ1(det,L,sigmaeq);            // Set up for 1D acquisition
```

See Also:

4.5.5 Gpulse_U

Usage:

```
#include <P_Gaussian.h>
gen_op Gpulse_U(gen_op& Ho, gen_op& Fxy,
                        int N, double ang, double tp, double fact)
```

Description:

The function ***Gpulse_U*** generates a propagator which will evolve a spin system under a Gaussian shaped pulse. The waveform consists of ***N*** steps, is of length ***tp*** (sec.), begins at intensity ***fact*** (decimal% of maximum), and will rotate magnetization on resonance by angle ***ang*** (degrees). The operator ***Fxy*** sets the pulse phase and selectivity. The operator ***Ho*** is the isotropic system Hamiltonian.

Return Value:

The function a single propagator (operator).

Example:

```
#include <P_Gaussian.h>
Acquire1D ACQ1(det,L,sigmaeq);            // Set up for 1D acquisition
```

See Also:

4.6 Auxiliary Functions

4.6.1 Gangle

Usage:

```
#include <P_Gaussian.h>
double Gangle(double gamB1, double tau, int N, double fact=0.025)
```

Description:

The function ***Gangle*** returns the Gaussian shaped pulse rotation angle on resonance in degrees. The Gaussian waveform is based on the input field strength ***gamB1***in Hertz, the pulse length ***tau*** in seconds, the number of steps ***N***, and the decimal percent of maximum ***fact*** at the end points.

Return Value:

The function returns a double precision number.

Example:

```
#include <P_Gaussian.h>
double gamB1 = 50;                          // Set gamma*B1 to 50 Hz
int nstp = 1000;                            // Set number of steps to 500
double tp = 0.30;                           // Set the pulse length to 30 ms
cout << "\nPulse Angle: "                   // Output the results of Gangle
     << Gangle(gamB1,tp,nstp)               // (roughly 270 degrees in this case)
     << " Degrees";
```

See Also:

4.6.2 GgamB1

Usage:

```
#include <P_Gaussian.h>
double GgamB1 (double angle, double tau, int N, double fact=0.025)
```

Description:

The function ***GgamB1*** returns the Gaussian shaped pulse rf-strength needed to attain an on-resonance rotation angle of ***angle*** degrees. The Gaussian waveform is based on the rotation angle ***angle*** in degrees, the pulse length ***tau*** in seconds, the number of steps ***N***, and the decimal percent of maximum ***fact*** at the end points.

Return Value:

The function returns a double precision number.

Example:

```
#include <P_Gaussian.h>
```

```
double tp = 0.01;                              // Set pulse length to 10 ms
int N = 1001;                                  // Set number of steps to 1001
double ang = 270.;                             // Set the angle to 270 degrees
double fact = 0.025;                           // Set cutoff to 2.5% at endpoints
double GB1 = GgamB1(ang,tp,N,fact);            // Get the needed field strength
cout << "\nSet Field to " << GB1 << " Hz";     // Output calculated field (~160Hz)
```

See Also:

4.6.3 Gtime

Usage:

```
#include <P_Gaussian.h>
double Gangle(double angle, double gamB1, int N, double fact=0.025)
```

Description:

The function ***Gtime*** returns the Gaussian shaped pulse length in seconds required to attain an on-resonance rotation angle of ***angle*** degrees. The Gaussian waveform is based on the rotation angle ***angle*** in degrees, the input field strength ***gamB1***in Hertz, the number of steps ***N***, and the decimal percent of maximum ***fact*** at the end points.

in degrees.

Return Value:

The function returns a double precision number.

Example:

```
#include <P_Gaussian.h>
double angle = 270.;                  // Set pulse angle to 270 degrees
int nstp = 1001;                      // Set number of steps to 1001
double gamB1 = 50.;                   // Set the field strength to 50 Hz
cout << "\nPulse Length: "            // Output the results of Gtime
     << Gtime(ang,gamB1,nstp)         // (roughly 32 ms in this case)
     << " Seconds";
```

See Also:

4.6.4 GNvect

Usage:

```
#include <P_Gaussian.h>
double Gangle(double gamB1, double tau, int N, double fact=0.025)
```

Description:

The function ***Gangle*** returns the Gaussian shaped pulse rotation angle on resonance in degrees.

Return Value:

The function is void, it will alter the input data block.

Example:

#include <P_Gaussian.h>

See Also:

4.6.5 Gvect

Usage:

```
#include <P_Gaussian.h>
double Gangle(double gamB1, double tau, int N, double fact=0.025)
```

Description:

The function ***Gangle*** returns the Gaussian shaped pulse rotation angle on resonance in degrees.

Return Value:

The function is void, it will alter the input data block.

Example:

#include <P_Gaussian.h>

See Also:

4.6.6 GIntvec

Usage:

```
#include <P_Gaussian.h>
double Gangle(double gamB1, double tau, int N, double fact=0.025)
```

Description:

The function ***Gangle*** returns the Gaussian shaped pulse rotation angle on resonance in degrees.

Return Value:

The function is void, it will alter the input data block.

Example:

#include <P_Gaussian.h>

See Also:

4.7 Input/Output Functions

4.7.1 Ghistogram

Usage:

```
#include <P_Gaussian.h>
double Gangle(double gamB1, double tau, int N, double fact=0.025)
```

Description:

The function ***Gangle*** returns the Gaussian shaped pulse rotation angle on resonance in degrees.

Return Value:

The function is void, it will alter the input data block.

Example:

```
#include <P_Gaussian.h>
```

See Also:

4.7.2 ask_Gpulse

Usage:

```
#include <P_Gaussian.h>
double Gangle(double gamB1, double tau, int N, double fact=0.025)
```

Description:

The function ***Gangle*** returns the Gaussian shaped pulse rotation angle on resonance in degrees.

Return Value:

The function is void, it will alter the input data block.

Example:

```
#include <P_Gaussian.h>
```

See Also:

4.7.3 read_Gpulse

Usage:

```
#include <P_Gaussian.h>
double Gangle(double gamB1, double tau, int N, double fact=0.025)
```

Description:

The function ***Gangle*** returns the Gaussian shaped pulse rotation angle on resonance in degrees.

Return Value:

The function is void, it will alter the input data block.

Example:

#include <P_Gaussian.h>

See Also:

4.7.4 <<

Usage:

```
#include <P_Gaussian.h>
ostream& operator << (ostream& ostr, acquire1D& ACQ1)
```

Description:

The operator << adds the acquisition specified as an argument ***ACQ1*** to the output stream ***ostr***. The format will as follows:

non-zero points out of # possible

Dwell time: # (if available)

A[i], B[i] pairs

Hilbert space basis.

Return Value:

None.

Example(s):

#include <P_Gaussian.h>

See Also:

4.8 Auxiliary Functions

4.8.1 size

Usage:

```
#include <P_Gaussian.h>
int acquire1D::size( )
```

Description:

The function ***size*** returns the current dimension over which the index ***p*** will sum in the generalized ***class acquire1D*** acquisition equation

$$\langle Op(t_k)\rangle = \sum_{p}^{size} A_p[B_p]^k$$

Return Value:

The function returns an integer.

Example:

```
#include <P_Gaussian.h>
```

See Also:

4.8.2 size

Usage:

```
#include <P_Gaussian.h>
int acquire1D::size( )
```

Description:

The function ***size*** returns the current dimension over which the index ***p*** will sum in the generalized ***class acquire1D*** acquisition equation

$$\langle Op(t_k)\rangle = \sum_{p}^{size} A_p[B_p]^k$$

Return Value:

The function returns an integer.

Example:

```
#include <P_Gaussian.h>
```

See Also:

4.9 Description

4.9.1 Introduction

The module ***P_Gaussian*** provides functions which pertain to Gaussian shaped pulses in NMR simulations. These functions either return propagators which evolve the density operator or they act on the density operator directly.

4.9.2 Analog Mathematical Basis

The Gaussian function is formally given by

$$G(t) = e^{[-(t-t_0)^2/2\sigma^2]} \tag{19-1}$$

where σ is the standard deviation and relates the linewidth at half-height by the relationship

$$t_{1/2} = \sqrt{8ln(2)}\sigma = (2.35482)\sigma \ . \tag{19-2}$$

The Gaussian function maximizes to 1 at $t = t_o$ and is zero at $t = \pm\infty$. A plot of this function would be

Analog Gaussian Plot

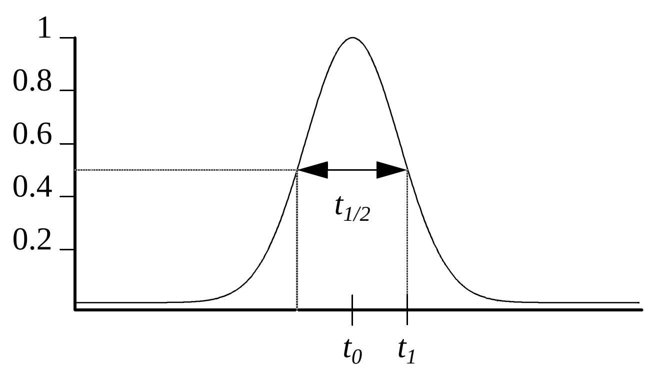


Figure 19-3 A Gaussian function depicting the peak maximum at $t = t_o$ and the linewidth at half-height. This function maximum is 1 and the end points tend to zero at $t = \pm\infty$. This plot was produced from the program Gplot.cc given at the end of this chapter.

The relationship between σ and $t_{1/2}$ is derived as follows.

$$\frac{f(t_1)}{f(t_o)} = \frac{0.5}{1} = \frac{exp[-(t_1-t_0)^2/(2\sigma^2)]}{exp[-(t_0-t_0)^2/(2\sigma^2)]}$$

$$\frac{1}{2} = exp\left[\frac{-(t_1 - t_0)^2}{2\sigma^2}\right] = exp\left[\frac{-(0.5t_{1/2})^2}{2\sigma^2}\right]$$

$$ln(0.5) = \frac{-(0.5t_{1/2})^2}{2\sigma^2} \qquad 2ln(2) = \frac{(0.5t_{1/2})^2}{\sigma^2} \qquad \frac{1}{2}t_{1/2} = \sigma\sqrt{2ln(2)}$$

$$t_{1/2} = \sqrt{8ln(2)}\sigma = (2.35482)\sigma \tag{19-3}$$

Because the Gaussian define above maximizes to a value of 1 whereas we can change its linewidth by altering the standard deviation, the integrated area under the curve varies with σ. If desired, a normalization factor may be placed in front of the Gaussian so that it's integrated intensity is 1 rather that its maximum height. This is done by multiplication by $1/N$ where N is given by

$$N = \int_0^{\infty} G(t)dt = \sqrt{2\pi}\sigma \tag{19-4}$$

a value that can be obtained as follows.

$$N^2 = \int_0^{\infty} exp\left[\frac{-(x-x_0)^2}{2\sigma^2}\right]dx\int_0^{\infty} exp\left[\frac{-(y-y_0)^2}{2\sigma^2}\right]dy = \int_0^{\infty}\int_0^{2\pi} exp\left[\frac{-r^2}{2\sigma^2}\right]rdrd\theta$$

$$= 2\pi\int_0^{\infty} exp\left[\frac{-r^2}{2\sigma^2}\right]rdr = 2\pi\sigma^2\int_0^{\infty} e^{-u}du = 2\pi\sigma^2$$

4.9.3 Discrete Mathematical Basis

A computer representation of any waveform, such as the Gaussian function, must be done using discrete mathematics. The waveform will be repersented by a specified number of points, N, and the previous analog function can be adjusted to evaluate at each point according to

$$G_i = e^{[-(i-i_0)^2/2\sigma^2]} \tag{19-5}$$

However, we will demand a few modifications of this formula to make it suitable for defining a pulse shape in NMR. Keep in mind that a pulse programmer in a spectrometer is limited to the same discrete mathematics. The applied pulse wave form is only a discrete approximation to the true pulse function. We will tailor our discrete formula according to the following two conditions.

1.) The Gaussian maximum will be centered in the middle of the Gaussian points. Since the waveform will be applied at a specified time in a pulse sequence, we can move the center of the pulse to any point in the sequence. For an array of N points, the center is given by[1]

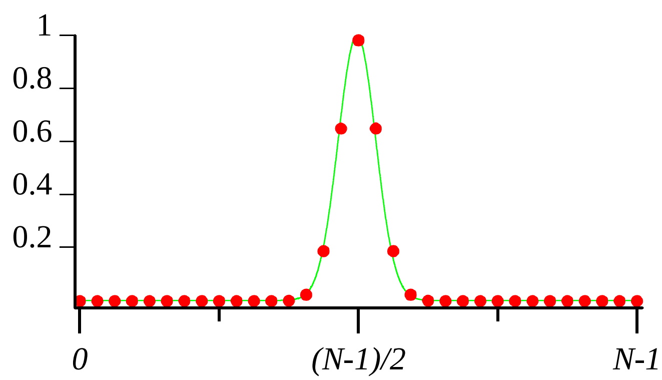
$$(N-1)/2 \tag{19-6}$$

and so our working equation becomes

$$G_i = e^{[-[i-(N-1)/2]^2/2\sigma^2]} = e^{[([2i-(N-1)]^2\ln(fact))/2\sigma^2]} \tag{19-7}$$

In this formula i is a non-negative integer, the point (or step) increment. The units on sigma must also be points but it need not be integer. Below is a plot of G_i using $N = 33$ and $\sigma = 1.1$ overlaid is an analog Gaussian.

Analog vs. Discrete Gaussian Plot



2.)Unlike the analog Gaussian, here we would like to set the value of σ, or equivalently the Gaus-

1. This if or C indexing, the first point indexed as 0 and the last point as N-1. Not that there may not actually be an evaluated point in the center. If N is even then the center will lie between two points of the discrete waveform.

sian spread, such that the intensity at the first point is a specified percentage of the maximum. Recall that a true Gaussian only approaches zero at an infinite distance away from the maximum. One would then need infinite time to attain a true Gaussian pulse shape, so we settle for shorter pulses by just truncating the Gaussian at some specified minimum height (~1%).

Rather than set the Gaussian width in terms of a σ value[1], a more appropriate choise would be to just specifiy the end-point intensity relative the the function maximum. In building Gaussian shaped pulses this is important because normally the initial Gaussian intensity is specified and should not be set to zero. We must choose a cutoff value which indicates the initial and final intensities of the discrete function based on a set percentage of the maximum intensity, $fact$, where $fact$ is the decimal form of a percent (e.g. 2% maximum is $fact = 0.02$). We can see how this will affect our discrete formula, or equivalently, what value of σ is required, by looking at either the first (i=0) or last (i=N-1) point.

$$fact = e^{[-[i-(N-1)/2]^2/2\sigma^2]}\Big|_{i=0} = e^{[-[i-(N-1)/2]^2/2\sigma^2]}\Big|_{i=N-1} \tag{19-8}$$

So that

$$fact = e^{-[(N-1)/2]^2/2\sigma^2} = e^{-(N-1)^2/8\sigma^2} \tag{19-9}$$

where $fact \in [0,1]$. Back solving this for the standard deviation produces

$$\sigma = (N-1)/\sqrt{-8\ln fact} \tag{19-10}$$

Putting this back into our original discrete Gaussian amplitude using

$$1/2\sigma^2 = -\ln(fact)/[(N-1)/2]^2 \tag{19-11}$$

The discrete Gaussian equation is then

$$G_i = e^{\ln(fact)[2i-(N-1)]^2/(N-1)^2} \tag{19-12}$$

How the discrete function behaves is shown in the next figure.

1. That would allow for either most points to be zero by selecting a very small standard deviation, or for having the defined Gaussian almost constant by selecting a very large standard deviation.

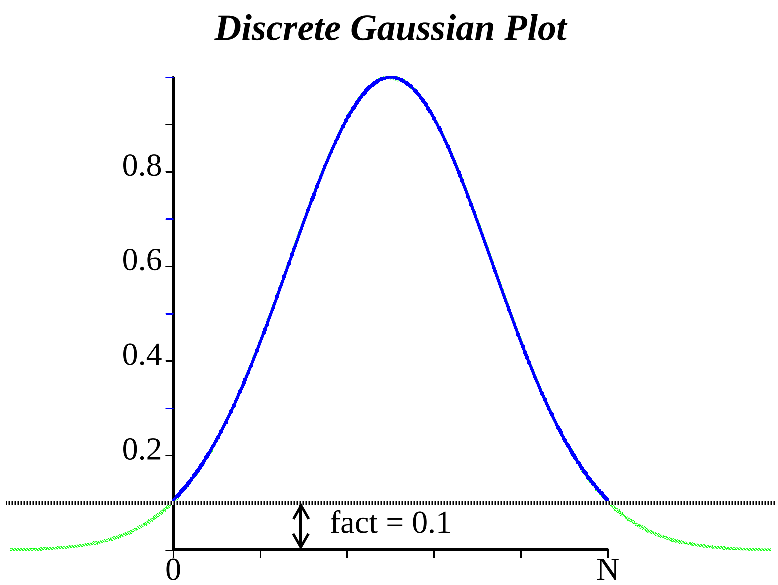


Figure 19-4 A discrete Gaussian function depicting the peak maximum at $(N-1)/2$. In this case the "linewidth" is set by the function intensity at the two endpoints, in decimal form percent of maximum peak height, fact.This plot was produced by program Gplot.cc given at the end of this chapter.

If we now check the Gaussian endpoints we find that

$$G_0 = G_{N-1} = fact$$

which is what we intended. Furthermore, the discrete Gaussian is symmetric, as can be proven by demonstrating that $G_{(N-1)-i} = G_i$.

$$G_{(N-1)-i} = exp\left[\frac{(2(N-1-i)-(N-1))^2\ln(fact)}{(N-1)^2}\right] = exp\left[\frac{(N-1-2i)^2\ln(fact)}{(N-1)^2}\right]$$

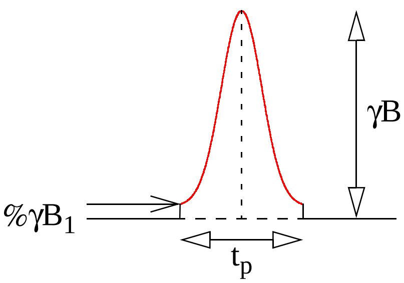
$$= exp\left[\frac{(2i-(N-1))^2\ln(fact)}{(N-1)^2}\right] = G_i$$

4.9.4 Discrete Pulse Mathematics

Having discussed the equations which apply to Gaussian functions, we turn our attention to construction of a Gaussian pulse. In this application the function merely defines the relative intensity of an applied rf-field in time. At each point in the discrete function, the rf intensity is adjusted to a new value and that value is maintained until the next point or the end of all points.

A Gaussian pulse is specified in part by a pulse length (t_p), a field strength at the maximum, (γB_1), and a percentage of this value that will be the rf intensity at the beginning of the pulse. This is depicted in the following figure.

Gaussian Pulse Parameters



Additionally, because instrument amplifiers cannot do an analog Gaussian intensity modulation, the pulse is broken up into a number of steps and this number also characterizes the pulse. We can readily depict this situation by drawing rectangles representing the rf-strengths during each point.

Gaussian Pulse Shape

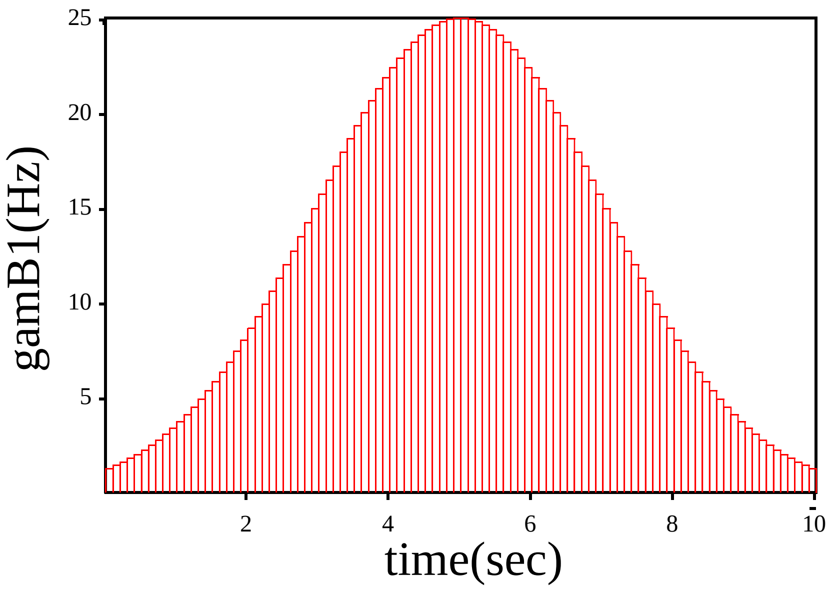


Figure 19-5 A Gaussian pulse wave form. The rf-field strength (gamB1) is discretely changed over a finite time increment based on the number of steps and the total length of the pulse. This plot was produced from the program Gplot.cc given at the end of this chapter.

Our plan is to maintain symmetric Gaussian waveforms, regardless of the number of point characterizing them. Examples are shown in the next figure.

Gaussian Pulse Symmetry

In practice there is a small delay between each step and each step will not be a true square wave as rf amplifiers cannot turn on and off instantaneously.

Now, the equation for the Gaussian intensity is given by

$$G_i = \gamma B_1 e^{\ln(fact)[2i-(N-1)]^2/(N-1)^2} \tag{19-13}$$

which is simply our previous formula multiplied by an rf-field strength γB_1. This strength is maintained for a specified time increment, Δt, where the total pulse length for the N steps is

$$t_p = N\Delta t \tag{19-14}$$

Note that the discrete Gaussian intensities, as written above, do not contain any time variables. Howevert the two are related because the "on-resonance" angle of rotation for any step is given by

$$\theta_i = G_i \times \Delta t \tag{20}$$

and the total "on-resonance" rotation due to the Gaussian pulse by

$$\theta_p = \sum_{i=0}^{N-1} G_{p,i} \times \Delta t = t_p \sum_{i=0}^{N-1} G_{p,i} \times \Delta t = \gamma B_1 t_p \sum_{i=0}^{N-1} G_i \tag{21}$$

Because the integral of the plain discrete Gaussian,

$$\sum_{i=0}^{N-1} G_i$$

depends on the number of steps taken, N, it is clear that the parameters

$$\theta_p \qquad t_p \qquad \gamma B_1 \qquad N$$

are related. Often the value of γB_1 is determined by setting the other three parameters according to

$$\theta_p \Big/ \left(t_p \sum_{i=0}^{N-1} G_i \right) = \gamma B_1 \tag{22}$$

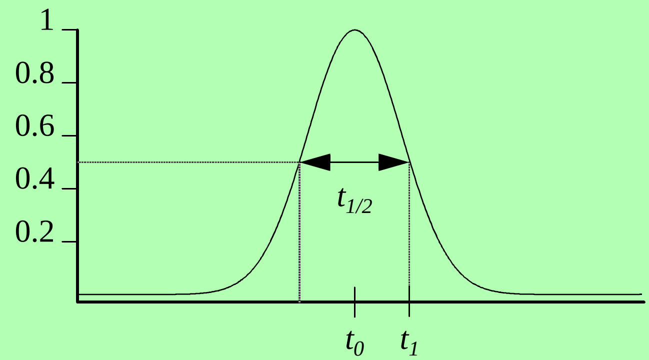
Gaussian Pulse Summary

Analog Gaussians

$$G(t) = exp\left[\frac{-(t-t_o)^2}{2\sigma^2}\right]$$

$$t_{1/2} = \sqrt{8ln(2)}\sigma$$

$$\int_0^{\infty} G(t)dt = \sqrt{2\pi}\sigma$$

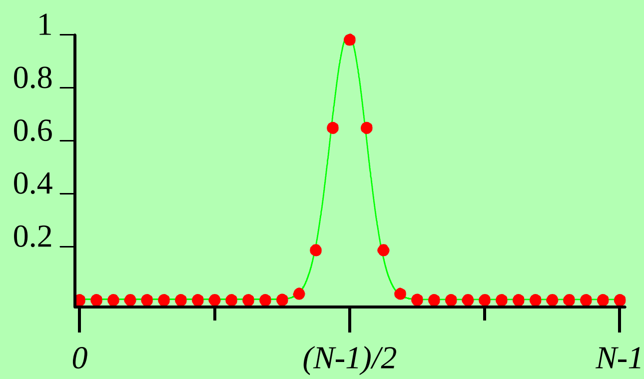


Discrete Gaussians

$$G_i = exp\left[\frac{-\left(i-\frac{1}{2}(N-1)\right)^2}{2\sigma^2}\right]$$

$$fact \in [0,1]$$

$$G_i\Big|_{\substack{max = 1 \\ min = fact}} = e^{\left[\frac{(2i-(N-1))^2 \ln(fact)}{(N-1)^2}\right]}$$

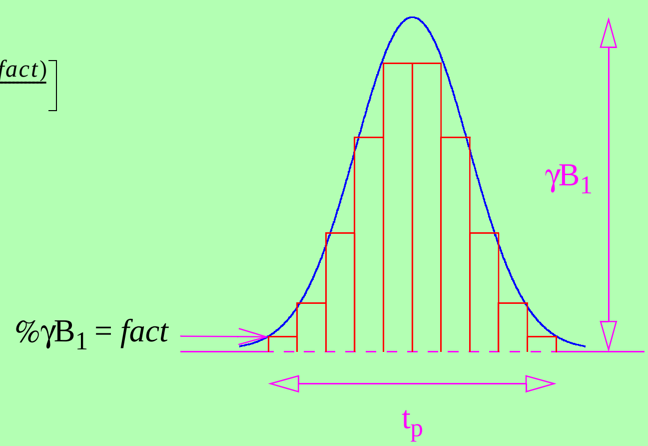


Discrete Gaussian Pulses

$$G_{p,i}\Big|_{\substack{max = \gamma B_1 \\ min = fact}} = \gamma B_1 e^{\left[\frac{(2i-(N-1))^2 \ln(fact)}{(N-1)^2}\right]}$$

$$t_p = N \times \Delta t$$

$$\gamma B_1 = \frac{\theta_p}{t_p \sum_{i=0}^{N-1} G_i}$$



4.9.5 Gaussian Pulses, No Relaxation

Without relaxation, each step i is given by the solution to the Liouville equation under a constant effective Hamiltonian in the rotating frame of the applied rf-field. In this case we can write

$$\hat{\underset{\sim}{\sigma}}_{i+1} = \hat{U}_i \hat{\underset{\sim}{\sigma}}_i U_i^{-1} \tag{22-1}$$

where $\hat{U}_i$ is a unitary propagator which evolves the system for time Δt under the Gaussian field strength G_i. The underscore tilde ~ is meant to denote the rotating frame of the applied rf-field. Individual propagators are generated from the effective Hamiltonian

$$\hat{U}_i = \exp(-i\hat{H}_{i,\,eff}\Delta t) \tag{22-2}$$

where

$$\hat{H}_{i,\,eff} = \hat{H}_0 - \Omega_{rf}\hat{F}_{z,\,i} + G_i\hat{F}_{x,\,y} \tag{22-3}$$

Starting with the initial density operator, $\hat{\underset{\sim}{\sigma}}_0$, at the end of N steps we will have

$$\hat{\underset{\sim}{\sigma}}_N = \left(\prod_{i=0}^{N-1} \hat{U}_i\right) \hat{\underset{\sim}{\sigma}}_0 \left(\prod_{i=0}^{N-1} \hat{U}_i\right)^{-1} \tag{22-4}$$

Note that since each the propagators evolves the density operator for a time Δt, at the end of the sequence the time will be the length of the applied shaped pulse

$$t_p = N\Delta t \tag{22-5}$$

We can define a Gaussian pulse propagator (no relaxation) to be

$$\hat{U}_{GP}(t_p, \gamma B_1, \Omega_{rf}, N) = \prod_{i=0}^{N-1} \hat{U}_i(\Delta t, \gamma B_1, \Omega_{rf}) = \hat{U}_{N-1} \ldots \hat{U}_1 \hat{U}_0 \tag{22-6}$$

Evolution under a Gaussian pulse is then given by

$$\hat{\underset{\sim}{\sigma}}(t_p + t_0) = \hat{U}_{GP}(t_p, \gamma B_1, \Omega_{rf}, N)\hat{\underset{\sim}{\sigma}}(t_0)\hat{U}_{GP}^{-1}(t_p, \gamma B_1, \Omega_{rf}, N) \tag{22-7}$$

4.9.6 Gaussian Pulses, With Relaxation

We shall now repeat the mathematical flow of the previous sections but account for relaxation in a rigorous fashion using WBR theory. In this case spin system evolution is given by

$$\underset{\sim}{\sigma}_{i+1} = \exp(-i\hat{\hat{L}}_i\Delta t)\Delta\underset{\sim}{\sigma}_i + \underset{\sim}{\sigma}_{i,ss} \tag{22-8}$$

where $\hat{\hat{L}}_i$ is the Liouvillian superoperator which dictates spin system evolutionn, $\Delta\hat{\hat{\underset{\sim}{\sigma}}}_i$ the difference density operator at point i, and $\hat{\hat{\underset{\sim}{\sigma}}}_{i,ss}$ the steady state at that same point. The difference density operator is given by

$$\Delta\hat{\hat{\underset{\sim}{\sigma}}}_i = \hat{\hat{\underset{\sim}{\sigma}}}_i - \hat{\hat{\underset{\sim}{\sigma}}}_{i,ss} \tag{22-9}$$

wheras the steady-state matrix itself is determined from

$$\hat{\underset{\sim}{\sigma}}_{i,ss} = \frac{\hat{\hat{\underset{\sim}{R}}}_i}{\hat{\underset{\sim}{L}}_i}\hat{\sigma}_{eq} \tag{22-10}$$

the superoperator $\hat{\hat{\underset{\sim}{R}}}_i$ containing all Liouvillian terms except those from the commutation Hamiltonian superoprator.

Because we plan to cycle through many steps in the application of our Gaussian pulse, it is convenient to rewrite equation (22-8) interms of superoperator propagators.

$$\underset{\sim}{\sigma}_{i+1} = \Gamma_i\underset{\sim}{\sigma}_i + \underset{\sim}{\sigma}_{i,ss} \tag{22-11}$$

4.9.7 Gaussian Pulse Equations

In this section we regroup the applicable equations regarding ***Gaussian Pulses***

Gaussian Shaped Pulse Equations

Overall

$$\langle Op(t)\rangle = Tr\{Op\cdot\sigma(t)\} = \langle Op^{\dagger}|\sigma(t)\rangle =$$

Unitary Transformation, Hilbert Space

$$\langle Op(t_k)\rangle = \sum_p A_p[B_p]^k = Tr\left\{Op\cdot \boldsymbol{U}^k\sigma(t_o)[\boldsymbol{U}^{-1}]^k\right\}$$

$$A_{\alpha\alpha'} = \langle\alpha|Op|\alpha'\rangle\langle\alpha'|\sigma(t_o)|\alpha\rangle$$

$$B_{\alpha\alpha'} = \langle\alpha'|\boldsymbol{U}|\alpha'\rangle\langle\alpha|[\boldsymbol{U}^{-1}]|\alpha\rangle$$

$$p = \alpha\alpha' \;\forall\; \langle\alpha|Op|\alpha'\rangle \neq 0$$

Expectation Value at Time t_k

$$\sigma(t_k) = \boldsymbol{U}^k\sigma(t_o)[\boldsymbol{U}^{-1}]^k$$

$$\boldsymbol{U} = e^{-i\boldsymbol{H}(\Delta t)}$$

$$\langle Op(t_k)\rangle = \sum_p A_p[B_p]^k$$

Non-Unitary Transformation, Liouville Space

$$\langle Op(t_k)\rangle = \sum_p A_p[B_p]^k = Tr\{Op\cdot\Gamma_{RP}^k\sigma(t_o)\}$$

$$A_{\alpha\alpha'} = \langle 1|Op^{\dagger}S|\alpha\alpha'\rangle\langle\alpha\alpha'|S^{-1}\sigma(t_o)|1\rangle$$

$$B_{\alpha\alpha'} = \langle\alpha\alpha'|\Lambda|\alpha\alpha'\rangle$$

$$\sigma(t_k) = \Gamma^k\sigma(t_o)$$

$$p = \alpha\alpha' \;\forall\; \langle 1|Op^{\dagger}S|\alpha\alpha'\rangle \neq 0$$

Redfield Theory, Liouville Space

$$\langle Op(t_k)\rangle = \sum_p A_p[B_p]^k + Tr\{Op\cdot\hat{\sigma}_{inf}\} = Tr\{Op\cdot[\Gamma^k(\sigma(t_o)-\hat{\sigma}_{inf})+\hat{\sigma}_{inf}]\}$$

$$A_{\alpha\alpha'} = \langle 1|Op^{\dagger}S|\alpha\alpha'\rangle\langle\alpha\alpha'|S^{-1}[\sigma(t_o)-\hat{\sigma}_{inf}]|1\rangle$$

$$B_{\alpha\alpha'} = \langle\alpha\alpha'|\Lambda|\alpha\alpha'\rangle$$

$$\sigma(t_k) = \Gamma^k\{(t_o)-\hat{\sigma}_{inf}\}+\hat{\sigma}_{inf}$$

$$p = \alpha\alpha' \;\forall\; \langle 1|Op^{\dagger}S|\alpha\alpha'\rangle \neq 0$$

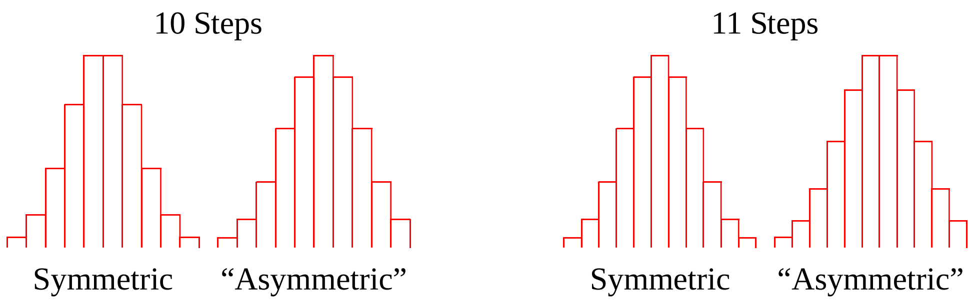
$$\Gamma = e^{-L\Delta t}$$

4.9.8 Final Notes

A discrete Gaussian function, that is to say a finite array of points with values related to a Gaussian distribution, will NOT be zero at its endpoints. Rather it will be some finite value the may become close to zero within the machine precision. In building Gaussian shaped pulses this is important because normally the initial Gaussian intensity is specified (say at 5% maximum) and should not be set to zero.

(23)

Additionally, because instrument amplifiers cannot do an analog Gaussian intensity modulation, the pulse is broken up into a number of steps and this number also characterizes the pulse. Examples are shown in the next figure. For a given number of steps, the Gaussian can be broken up symmetrically or “asymmetrically”. The former requires more sophisticated tracking of the individual step intensities but will in principle produce better excitation profiles. The latter is mathematically easier to generate and likely to be what is supplied with a spectrometer.



Typically one will take a number of steps ~10^3 so the differences between these two constructs becomes small. In practice there is a small delay between each step and the each step is not a true square wave as the amplifier does not turn on and off instantaneously.

4.10 Gaussian Pulse Parameters

4.10.1 Introduction

Gaussian pulses are often used in NMR as they can be tailored to be highly selective (i.e. covering a selected frequency range) with relatively small amplitude and phase distortions. In GAMMA, such pulses are treated as a special cases (rather than simply a general shaped pulse[1]) because the pulse shape symmetry allows for significant computational savings. The module ***P_Gaussian*** provides a variety of functions pertaining to Gaussian shaped pulses. Of interest here are the functions in ***P_Gaussian*** that either return propagators which evolve the density operator or act on the density operator directly.

4.10.2 Gaussian Pulse Parameters

A Gaussian pulse is specified in part by a pulse length (t_p), a field strength at the maximum, (γB_1), and a percentage of this value that will be the rf intensity at the beginning of the pulse ($\%\gamma B_1$). Additionally, because instrument amplifiers cannot do an analog Gaussian intensity modulation, the pulse is broken up into a number of steps (N). We can readily depict this situation in the following figure, using rectangles to represent the rf-strengths during each point.

Four Gaussian Pulse Parameters

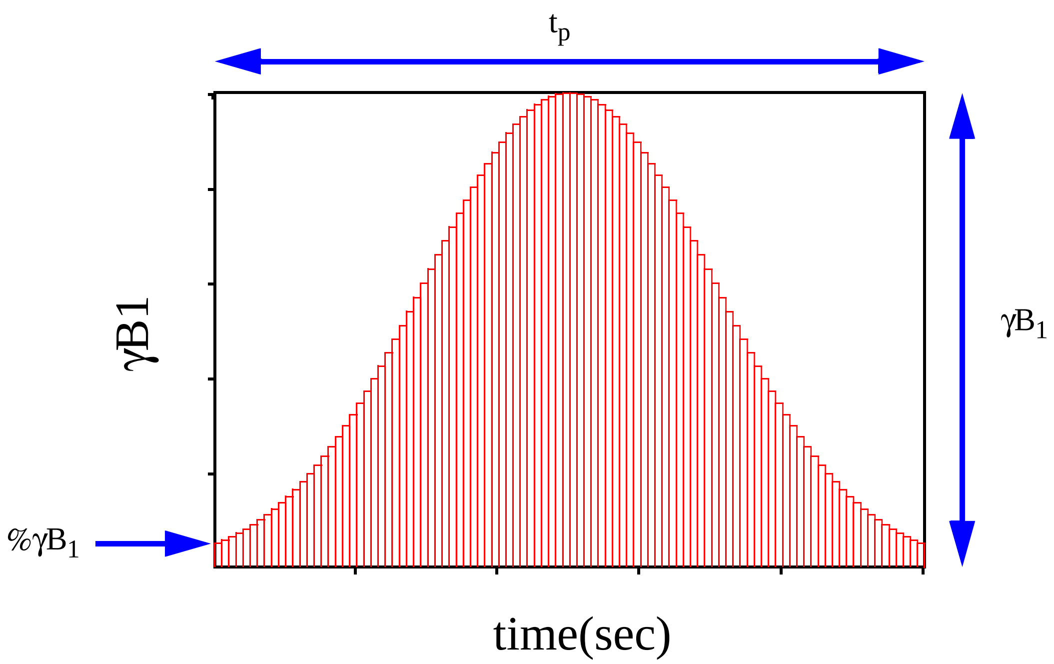


Figure 19-6 A Gaussian pulse wave form. The rf-field strength (gamB1) is discretely changed over

1. GAMMA users may construct arbitrary shaped pulses by simply supplying a vector containing the desired waveform and a few other pulse parameters. Look at the documentation regarding shaped pulses to see how that is accomplished. We recommend that you use the functions in the Gaussian pulse module when your programs require such pulses. They are easier to use and the routines faster computationally.

a finite time increment based on the number of steps and the total length of the pulse. The entire pulse waveform is determined from four parameters $\{t_p, \gamma B_1, N, \%\gamma B_1\}$. The program which produced this plot can be found in the documentation for the ***P_Gaussian*** module.

Three other parameters are required for a Gaussian pulse, the pulse offset (or carrier frequency), pulse phase, and the pulse channel. That makes a total of seven parameters for complete characterization of a Gaussian pulse in GAMMA. Indeed, use of GAMMA's Gaussian pulses is quite simple if the user has a clear understanding of the means by which the seven parameters are set in a the program. This will be the topic of the next sections.

4.10.3 Defining a Gaussian Pulse Directly

This task is accomplished by specifying the seven parameters that define a Gaussian pulse. For example, the following code will do (we will show how to make and apply the pulse later):

4.10.4 Defining a Gaussian Interactively

As in the last section, we need to specify the seven parameters that define a Gaussian pulse. In this case we need to have the program itself ask for the parameters as the program is run.

4.10.5 Defining a Gaussian Pulse in an External File

GAMMA provides a very simple means of defining a Gaussian pulse in an external "parameter" file. The parameter file is simply an ASCII file which contains parameters that a GAMMA Gaussian pulse type will recognize. A GAMMA parameter is a line in a file having the format

Name (type) : value - optional comment

There are 9 parameter names that will be recognized as defining a Gaussian pulse.

Table 1: Gaussian Pulse Parameters

Parameter Keyword	Assumed Units	Examples Parameter (Type) : Value - Statement
Gstps	none	Gstps (0) : 41 - Steps in Gaussian Pulse
Gang	degrees	Gang (1) : 90.0 - Gaussian Pulse Angle
Glen	seconds	Glen (1) : 0.1 - Gaussian Pulse length
Gcut	none	Gcut (1) : 0.025 - RF cutoff level (%GgamB1 -> 2.5%)
GW	Hz	GW (1) : 400.0 - Gaussian Pulse Carrier Frequency
Giso	none	Giso (2) : 1H - Gaussian Pulse Channel
GgamB1	Hz	GgamB1 (1) : 55. - Gaussian Pulse RF Field Strength

Table 1: Gaussian Pulse Parameters

Parameter Keyword	Assumed Units	Examples Parameter (Type) : Value - Statement
Gphi	degrees	Gphi (1) : 0.0 - Gaussian Pulse Phase
Gspin	none	Gspin (0) : 0 - Gaussian Spin Selectivity

Of course, there are only seven parameters necessary to completely characterize the pulse. Two of the above parameters are redundant. The first redundancy comes from the three parameters {Gang, Gtime, GgamB1}. Only two of the three need to be specified. The set {Gang, Gtime} will be used preferentially if all three are set in the parameter file. The second redundancy comes in the selectivity. Either {Gspin} or {Gwrf, Giso} Sets Selectivity. If Gspin is set it will be preferentially used and override any Giso and Gwrf settings. However, note that use of Gspin DOES NOT set the Gaussian pulse selectivity to only affect a particular spin (which is not experimentally possible for strongly coupled or overlapping spins). Rather, it sets the Gaussian pulse carrier to be at the spin Larmor frequency.

Two other points are worth mentioning. If one has a homonuclear spin system the selectivity does not have to be set. Thus, neither Giso nor Gspin need be set if the there is only 1 channel and the pulse does not need to be spin selective. Some simulations utilize multiple Gaussian pulses, so there is often a need to define more than one pulse in a parameter file. This may be accomplished by adding on a (#) to the Gaussian parameter name where # is simply an integer which is a pulse index. The set of parameters which define a pulse are then all set with the same number in their names and the number used in the GAMMA program which reads the parameters.

Note that pulse parameters can be mixed with other parameters in a single ASCII file. For example, you can readily include the Gaussian pulse definition in the same file that defines your spin system. The code and file would look something like

4.10.6 Constructing a Gaussian Pulse Propagator

d look something like

4.10.7 Example: Gaussian Pulse Profile

This example takes a single spin and applies a specified Gaussian pulse (90y). It repeats this process while moving the spins chemical shift through the frequency range over which the profile is desired. The response versus pulse offset for the spin is plotted, both with and without (magnitude) the phase information present.

Single Spin Gaussian Pulse Profile

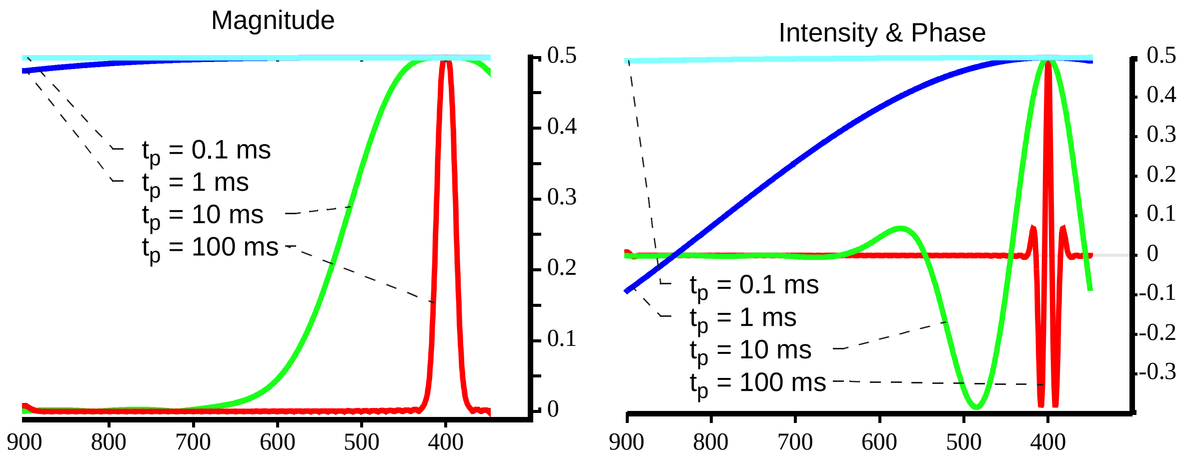


Figure 19-7 The plots were produced from successive runs of the program Gprofile2.cc on page 71. In all instances the Gaussian was applied at an offset of 400 Hz and the generated profiles constructed between 350 and 900 Hz with a block size of 1024 points. The program automatically sets the Gaussian to be a 90y pulse of 51 steps and an endpoint cutoff of 2.5% maximum intensity. Of minor interest is the small excitation produces near 900 Hz in the more selective Gaussian's. This is a consequence of using only 51 steps for the pulse shape and such effects disappear a the number of steps increase.

This simulation follows the general rule of thumb in NMR: short strong (hard) pulses promote even excitation over a broad frequency range and weak long (soft) pulses are selective in that they excite over a narrow frequency range. Notice that a second rule of thumb is also followed: Long pulses induce phase distortions off resonance. This is due primarily to the "dephasing" of the transitions during the time it takes to get the magnetization down into the xy-plane. At the end of a long 90 pulse, not all magnetization vectors are aligned along an axis perpendicular to the pulse because they have undergone precession during the pulse itself. In an ideal situation, the resulting spectrum would be phase adjusted using a 1st order phase correction. However, that assumes a linear response to the pulse which is not strictly the case - especially for strongly coupled spin systems and non-uniform pulse waveforms. So, phase distortions in a spectrum due to a long pulse can be difficult to remove by simple phase corrections.

4.10.8 Example: Gaussian 90 Pulse

This example takes a spin system and applies a specified Gaussian pulse (90y). It reads in both the spin system and the Gaussian pulse definition from a single GAMMA parameter file.

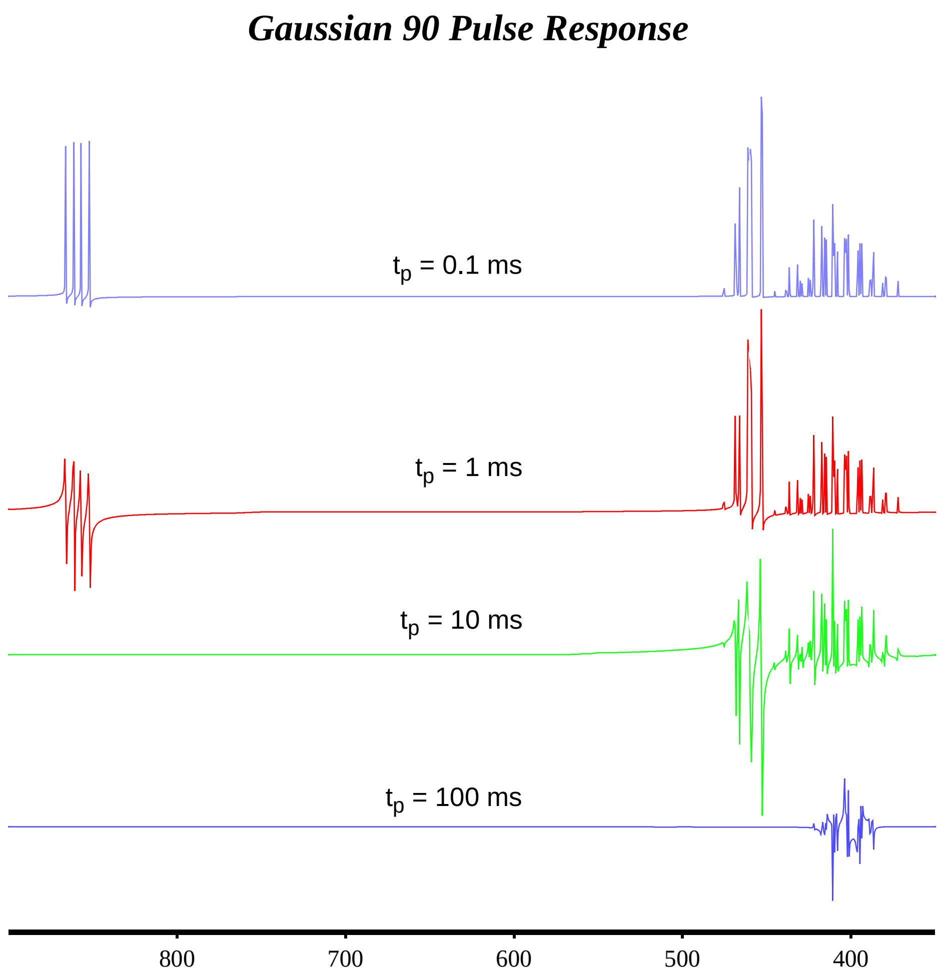


Figure 19-8 The plots were produced from successive runs of the program Gpulse0.cc on page 72. In all instances the program was given the parameter file GlutamicA.sys on page 73 which contains both a spin system definition and the Gaussian pulse parameters. Only the Gaussian pulse length was changed between the successive runs. The executable (a.out) was repeatedly invoked with the command "a.out GlutamicA.sys 350 900 1024 0.02" which set the plot range to span 350-900 Hz, the block size to 1K and the single quantum transition linewidths to .02 Hz.

Note that, although there is good selectivity with the 100 ms pulse, the inherent phases are terrible. In the next examples we shall attempt to minimize the phase errors by two different means. First, rather than using a 90 pulse we can attempt to use a Gaussian 270 pulse which has some self-refocusing properties which can reduce such problems[1]. Second we can attempt to perform a phase correction, either 1st order or using a single spin's pulse response to the pulse.

1. Or worsen them in strongly coupled spin systems!

4.10.9 Example: Gaussian 270 Pulse

This example takes a spin system and applies a specified Gaussian pulse (270y). It reads in both the spin system and the Gaussian pulse definition from a single GAMMA parameter file.

Gaussian 270 Pulse Response

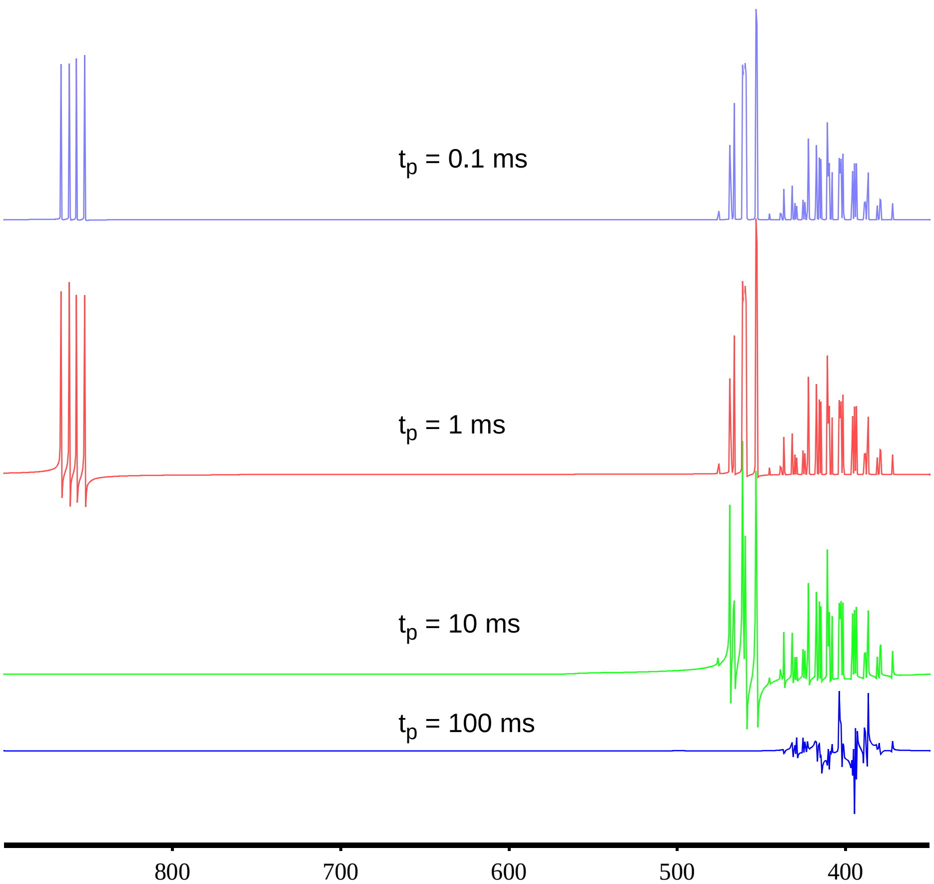


Figure 19-9 The plots were produced from successive runs of the program Gpulse1.cc on page 74. In all instances the program was given the parameter file GlutamicA2.sys on page 73 which contains both a spin system definition and the Gaussian pulse parameters. Only the Gaussian pulse length was changed between the successive runs. The executable (a.out) was repeatedly invoked with the command "a.out GlutamicA2.sys 350 900 1024 0.02" which set the plot range to span 350-900 Hz, the block size to 1K and the single quantum transition linewidths to 0.02 Hz.

In comparison with the previous simulation which used a 90 Gaussian pulse we see that there is some improvement in the phase behavior. However, there still remains quite a bit of dispersive nature to the multiplet with a 100 ms pulse.

4.11 Example: Gaussian Pulse, Profile Corrected

This example takes a spin system and applies a specified Gaussian pulse. It reads in both the spin system and the Gaussian pulse definition from a single GAMMA parameter file. In this case, it also creates a single spin response profile which displays how magnetization if affected by offset. Additionally, it adjusts the phases based on the single spin response - a rather robust phase correction.

Gaussian 270 Pulse Response, Profile Phase Corrected

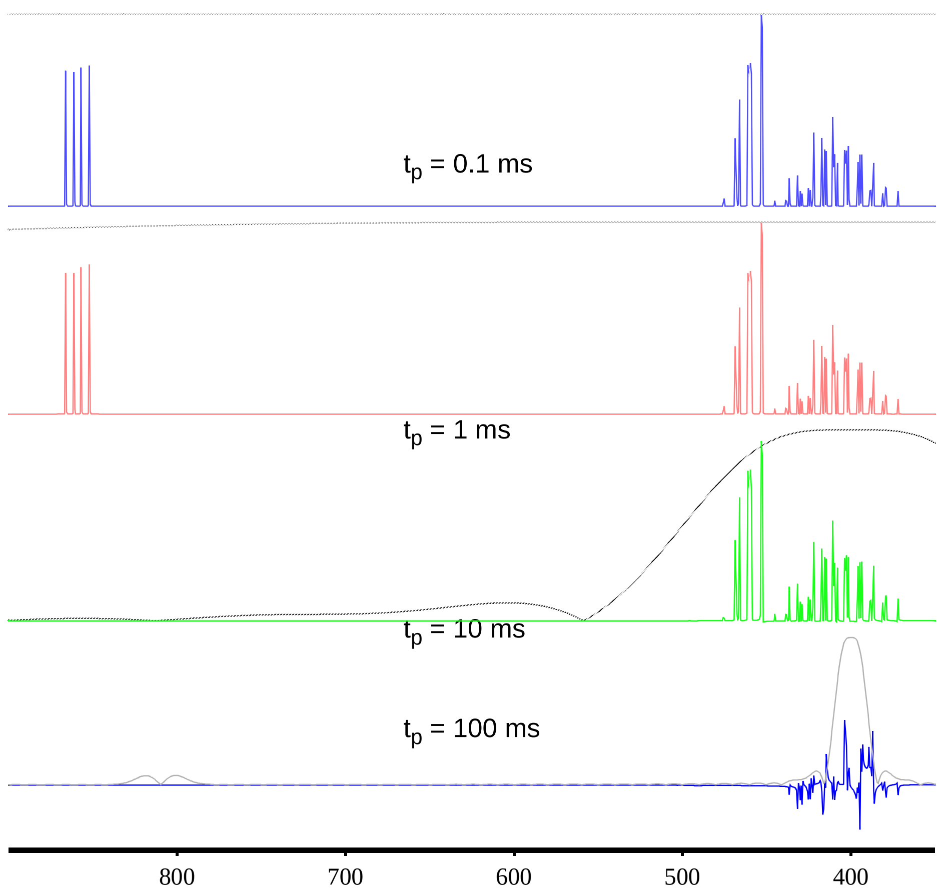


Figure 19-10 Plots produced from successive runs of Gpulcorr2.cc on page 75. The program was given the parameter file GlutamicA2.sys on page 73 which contains both a spin system definition and the Gaussian pulse parameters. Only the Gaussian pulse length was changed between the successive runs. The executable (a.out) was repeatedly invoked with the command “a.out GlutamicA2.sys 350 900 1024 0.02” which set the plot range to span 350-900 Hz, the block size to 1K and the single quantum transition linewidths to 0.02 Hz. The single spin profiles (magnitudes) are shown above the spectra in each case. (Note - a 90 Gaussian phase corrects better here!)

A quick comparison with the previous two simulations indicates that use of the single spin profile produces superior spectra. Unfortunately, this method would be time consuming and difficult to do experimentally. The 100 ms correction suffers from too few steps (41) and too high of end intensity (2.5%) characterizing the Gaussian, evident from the intensity near 800 Hz.

4.12 Example: Gaussian Pulse, Linear Phase Correction

Since we only have zero and first order phase corrections at our disposal on a spectrometer (without some fudging), it is perhaps worthwhile to examine our ability to use such a correction applied to the simulated spectra using selective Gaussians. Noting how difficult phase correction was in the previous example when the Gaussian was very long (selective), we expect linear phase correction to not perform very well.

Gaussian 90 Pulse Response, Standard Phase Correction

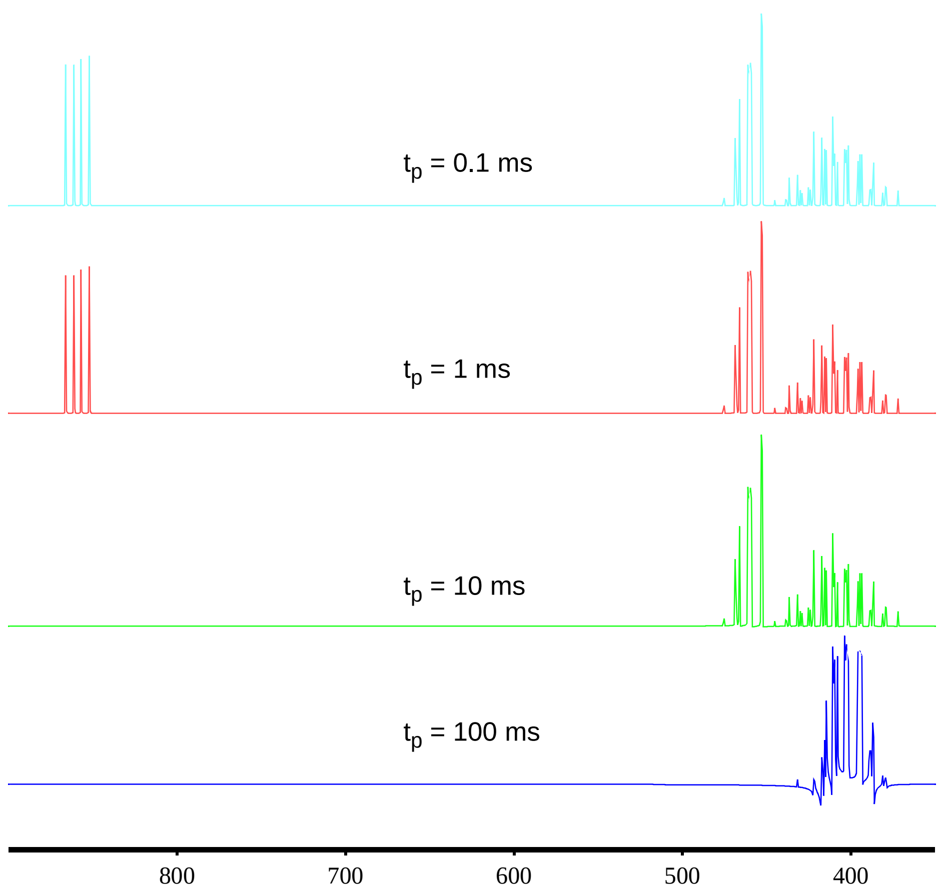


Figure 19-11 Plots produced from successive runs of Gpulpcorr2.cc on page 77. The program was given the parameter file slightly adjusted from GlutamicA2.sys on page 73 which contains both a spin system definition and the Gaussian pulse parameters. The Gaussian pulse length was changed between the successive runs. The pulse angle and phase were both set to 90. The executable (a.out) was repeatedly invoked with the command “a.out GlutamicA2.sys 350 900 1024 0.02” which set the plot range to span 350-900 Hz, the block size to 1K and the single quantum transition linewidths to 0.02 Hz.

Surprisingly, this type of phase correction works quite well. Again, I’ll point out that one cannot directly compare the 100 ms run here with the previous calculation as the 90 pulse in general seems to phase correct better than a 270 pulse in strongly coupled systems.

Gprofile2.cc

```
/* Gprofile2.cc ****************************************************************-*-c++-*-
**                                                                          **
** This program plots a Gaussian shaped pulse profile.  The response        **
** of a single spin to the shaped pulse is measured versus rf-offset.       **
** This version extends Gprofile1.cc by allowing for the plot to be         **
** asymmetric about 0 Hz.  Thuse the pulse can be applied at any            **
** frequency and the plot can between any two frequencies.                  **
**                                                                          **
** Author:     S.A. Smith                                                   **
** Date:       July 3 1996                                                  **
** Last Update: July 3 1996                                                 **
**                                                                          **
*****************************************************************************/

#include <gamma.h>                              // Include all of GAMMA

main(int argc, char* argv[])
  {
  cout << "\n\n\t\t\t   GAMMA 1D NMR Simulation Program";
  cout << "\n\t\t\tGaussian Pulse Profile, No Relaxation\n\n";
//            Set Gaussian Pulse Parameters

// (Set As A 90y Pulse Of 51 Steps, No Offset, No Phase, 2.5% Cutoff)

  int qn = 1;                                   // Query value
  Gpuldat Gdata;                                // For Gaussian pulse params
  double tp;                                    // Gaussian pulse length
  query_parameter(argc, argv, qn++,             // Read in this value
     "\n\tGaussian Pulse Length (sec)? ", tp);
  double Wrf;                                   // Gaussian pulse offset
  query_parameter(argc, argv, qn++,             // Read in this value
     "\n\tGaussian Pulse Offset(Hz)? ", Wrf);
  Gdata.N = 51;                                 // Set pulse steps
  Gdata.Wrf = Wrf;                              // Set pulse offset
  Gdata.Iso = String("1H");                     // Set pulse selectivity
  Gdata.tau = tp;                               // Set the pulse length
  Gdata.fact = 0.025;                           // Set pulse cutoff (2.5%)
  Gdata.phi = 0.0;                              // Set pulse phase
  Gdata.gamB1 = GgamB1(90.0, tp, 51, 0.025);    // Set the pulse strength
  print_Gpulse(cout, Gdata);                    // Print Gaussian pulse params
//                          Set the Profile Parameters

  int npts;                                     // Number of points
  query_parameter(argc, argv, qn++,             // Read in this value
          "\n\tProfile Block Size? ", npts);
  double Flow, Fhigh;                           // Profile frequency limits
  query_parameter(argc, argv, qn++,             // Read in this value
    "\n\tProfile Plotted Low Frequency (Hz)? ", Flow);
  query_parameter(argc, argv, qn++,             // Read in this value
    "\n\tProfile Plotted High Frequency (Hz)? ", Fhigh);
  double delW = (Fhigh-Flow)/double(npts -1);       // Frequency increment
//                          Set the Unchanging Entities

  spin_system sys(1);                               // A single proton
  gen_op D = Fm(sys);                      // Detect F-
  gen_op sigma0 = sigma_eq(sys);               // System at equilibrium
  gen_op Fxy = Fy(sys);                             // RF field Ham. (rot. fr.)
  gen_op wFz = complex(Gdata.Wrf)*Fz(sys);          // RF Offset (rot. fr.)
//                              Set the Changing Entities

  matrix mx;                                        // Matrix for transitions
  gen_op Heff;                                      // Effective Hams
  gen_op sigmap;                                     // Prepared density oper
  gen_op UGauss;
  gen_op H;
  complex z;
//                          Perform The Simulation

  row_vector profile(npts), profnorm(npts);         // This for profile & magnitude
  double offset = Flow;
  for(int i=0; i<npts; i++)
    {
    sys.shift(0, offset);                           // Shift relative to pulse
    H = Ho(sys) + wFz;                              // H in rotating frame
    UGauss = Gpulse_U(H, Fxy, Gdata);               // Gaussian pulse propagator
    sigmap = evolve(sigma0, UGauss);                // Evolve under the pulse
    z = trace(D, sigmap);                           // Get xy-magnetization
    profile.put(z,i);                               // Store magnetization at i
    profnorm.put(norm(z),i);                        // Store magnetization at i
    offset += delW;                                 // Increment the offset
    }

  cout << "\n\n";                                   // Keep screen nice
  cout.flush();                                     // Print all before gnuplot
  GP_1D("profile.gnu", profile,0,Flow,Fhigh);       // Spectrum out in gnuplot
  GP_1D("profnorm.gnu", profnorm,0,Flow,Fhigh);     // Magnitudes out in gnuplot
  ofstream gnuload("gnu.dat");                      // File of gnuplot commands
  gnuload << "set data style line\n";               // Set 1D plots to use lines
  gnuload << "plot \"profile.gnu\"";                // Plot the magnetization response
  gnuload << ", \"profnorm.gnu\"\n";                // Plot the magnitization magnitude
  gnuload << "pause -1 \'<Return> To Exit \n";      // Pause before quitting gnuplot
  gnuload << "exit\n";                              // Exit gnuplot
  gnuload.close();                                  // Close gnuplot command file
  system("gnuplot \"gnu.dat\"\n");                  // This actually does plot to screen

  cout << "\n\n";                                   // Keep the screen nice
  }
```

Gpulse0.cc

```
/* Gpulse0.cc *****************************************************************-*-c++-*-
**                                                                          **
**       NMR 1D Simulator Using A Gaussian Shaped Pulse                     **
**                                                                          **
**  This program is an automated 1D NMR spectral simulator using            **
**  shaped Gaussian pulses.  It runs inteactively, asking the user to       **
**  supply a parameter file filename as well as plot parameters.  The       **
**  system input is simply pulsed by the specified Gaussian.                **
**                                                                          **
**  This program does not include relaxation effects.  Also, the plot       **
**  is immediate in gnuplot, so this will die if gnuplot is that            **
**  program is not accessible. Finally, I made the y-axis the default       **
**  axis for the pulse (I don't know why anymore....) so that a 0 phase     **
**  for input results in absorption on resonance using F- to detect, if     **
**  the pulse is 90 that is.                                                **
**                                                                          **
** Author:    S.A. Smith                                                    **
** Date:      7/2/96                                                        **
** Update:    7/2/96                                                        **
** Version:   3.5                                                           **
**                                                                          **
******************************************************************************/

#include <gamma.h>                                  // Include GAMMA

main (int argc, char* argv[])

  {
  cout << "\n\n\t\t\tGAMMA 1D NMR Simulation Program";
  cout << "\n\t\t\tGaussian Pulses, No Relaxation\n\n";
//                   Read in System and Pulse Parameters

  int qn = 1;
  String filein;                                     // Input system file name
  query_parameter(argc, argv, qn++,                  // Get system file name
       "\n\tInput Parameter File? ", filein);
  sys_dynamic sys;                                   // A spin system
  sys.read(filein);                                  // Read in the system
  Gpuldat Gdata = read_Gpulse(filein, sys);          // Read Gaussian pulse params
  print_Gpulse(cout, Gdata);                         // Print Gaussian pusle params
  sys.offsetShifts(Gdata.Wrf);                       // Center system at pulse
//                Determine Isotope Detection Type, Set Variables

 String IsoD;
 query_isotope(sys, IsoD);                           // Get the detection isotope
 gen_op sigma = sigma_eq(sys);                       // Set density matrix equilibrium
 gen_op H = Ho(sys);                                 // Isotropic Hamiltonian
 gen_op detect = Fm(sys, IsoD);                      // Set detection operator to F-
 acquire1D ACQ(detect, H, 1.e-3);                    // No relaxation during acquisition (Ho)
 gen_op Fxy = Fy(sys, Gdata.Iso);                    // For Gaussian RF Hamiltonian
 gen_op UGauss = Gpulse_U(H,Fxy,Gdata);              // Gaussian pulse propagator
//                        Apply The Pulse Sequence

 sigma = evolve(sigma, UGauss);                      // Apply Gaussian pulse
 matrix mx = ACQ.table(sigma);                       // Set up 1D acquisition
//                        Set The Plotting Parameters

 double Fstart, Fend;                                // Number of points in FID
 query_parameter(argc, argv,                         // Get number of points
   qn++, "\n\tPlot Starting Frequency? ", Fstart);
 query_parameter(argc, argv,                         // Get number of points
   qn++, "\n\tPlot Final Frequency? ", Fend);
 int N;
 query_parameter(argc, argv,                         // Get number of points
   qn++, "\n\tPlot Points? ", N);
 double lwhh;                                        // Half-height linewidth
 query_parameter(argc, argv,                         // Get number of points
   qn++, "\n\tPlot Linewidths? ", lwhh);
 offset(mx, Gdata.Wrf, lwhh, 1);                     // Set input w offset, lwhh
//                        Construct Plot and Draw

 row_vector data=ACQ.F(mx,N,Fstart,Fend,1.e-3);// Frequency acquisition
 GP_1D("Gauss.gnu", data, 0 , Fstart, Fend);         // Output in gnuplot
 GP_1Dplot("gnu.dat", "Gauss.gnu");                  // Interactive 1D gnuplot
 cout << "\n";
 }
```

GlutamicA.sys

SysName	(2) : Glutamic	- System Name (glutamic acid)
NSpins	(0) : 5	- Number of Spins in the System
Iso(0)	(2) : 1H	- Spin Isotope Type
Iso(1)	(2) : 1H	- Spin Isotope Type
Iso(2)	(2) : 1H	- Spin Isotope Type
Iso(3)	(2) : 1H	- Spin Isotope Type
Iso(4)	(2) : 1H	- Spin Isotope Type
PPM(0)	(1) : 4.295	- Chemical Shifts in PPM
PPM(1)	(1) : 2.092	- Chemical Shifts in PPM
PPM(2)	(1) : 1.969	- Chemical Shifts in PPM
PPM(3)	(1) : 2.314	- Chemical Shifts in PPM
PPM(4)	(1) : 2.283	- Chemical Shifts in PPM
J(0,1)	(1) : 4.6	- Coupling Constants in Hz
J(0,2)	(1) : 9.5	- Coupling Constants in Hz
J(0,3)	(1) : 0.0	- Coupling Constants in Hz
J(0,4)	(1) : 0.0	- Coupling Constants in Hz
J(1,2)	(1) : -14.7	- Coupling Constants in Hz
J(1,3)	(1) : 7.3	- Coupling Constants in Hz
J(1,4)	(1) : 7.3	- Coupling Constants in Hz
J(2,3)	(1) : 7.3	- Coupling Constants in Hz
J(2,4)	(1) : 7.3	- Coupling Constants in Hz
J(3,4)	(1) : -14.6	- Coupling Constants in Hz
Omega	(1) : 200	- Field Strength MHz (1H based)

Gaussian Pulse Definitions

Note1: Two of {Gang, Gtime, GgamB1} used, {Gang, Gtime} preferentially
Note2: Either {Gspin} or {Gwrf, Giso} Sets Selectivity, {Gspin} preferentially
Note3: {Giso} Need not be set in a homonuclear system

Gstps	(0) : 41	- Steps over the Gaussian pulse
Gang	(1) : 90.0	- Pulse angle (degrees)
Glen	(1) : .1	- Pulse length (seconds)
Gcut	(1) : 0.025	- RF cutoff level (%GgamB1 -> 2.5%)
GW	(1) : 400.0	- Frequency at which to apply pulse
Giso	(2) : 1H	- Channel on which to apply pulse
Gphi	(1) : 0.0	- Phase on which to apply pulse
#GgamB1	(1) : 55.	- RF field strength (Hz)

GlutamicA2.sys

SysName	(2) : Glutamic	- System Name (glutamic acid)
NSpins	(0) : 5	- Number of Spins in the System
Iso(0)	(2) : 1H	- Spin Isotope Type
Iso(1)	(2) : 1H	- Spin Isotope Type
Iso(2)	(2) : 1H	- Spin Isotope Type
Iso(3)	(2) : 1H	- Spin Isotope Type
Iso(4)	(2) : 1H	- Spin Isotope Type
PPM(0)	(1) : 4.295	- Chemical Shifts in PPM
PPM(1)	(1) : 2.092	- Chemical Shifts in PPM
PPM(2)	(1) : 1.969	- Chemical Shifts in PPM
PPM(3)	(1) : 2.314	- Chemical Shifts in PPM
PPM(4)	(1) : 2.283	- Chemical Shifts in PPM
J(0,1)	(1) : 4.6	- Coupling Constants in Hz
J(0,2)	(1) : 9.5	- Coupling Constants in Hz
J(0,3)	(1) : 0.0	- Coupling Constants in Hz
J(0,4)	(1) : 0.0	- Coupling Constants in Hz
J(1,2)	(1) : -14.7	- Coupling Constants in Hz
J(1,3)	(1) : 7.3	- Coupling Constants in Hz
J(1,4)	(1) : 7.3	- Coupling Constants in Hz
J(2,3)	(1) : 7.3	- Coupling Constants in Hz
J(2,4)	(1) : 7.3	- Coupling Constants in Hz
J(3,4)	(1) : -14.6	- Coupling Constants in Hz
Omega	(1) : 200	- Field Strength MHz (1H based)

Gaussian Pulse Definitions

Note1: Two of {Gang, Gtime, GgamB1} used, {Gang, Gtime} preferentially
Note2: Either {Gspin} or {Gwrf, Giso} Sets Selectivity, {Gspin} preferentially
Note3: {Giso} Need not be set in a homonuclear system

Gstps	(0) : 41	- Steps over the Gaussian pulse
Gang	(1) : 270.0	- Pulse angle (degrees)
Glen	(1) : .0001	- Pulse length (seconds)
Gcut	(1) : 0.025	- RF cutoff level (%GgamB1 -> 2.5%)
GW	(1) : 400.0	- Frequency at which to apply pulse
Giso	(2) : 1H	- Channel on which to apply pulse
Gphi	(1) : 270.0	- Phase on which to apply pulse
#GgamB1	(1) : 55.	- RF field strength (Hz)

Gpulse1.cc

```
/* Gpulse1.cc ******************************************************************-*-c++-*-
**                                                                                      **
**        NMR 1D Simulator Using A Gaussian Shaped Pulse                               **
**                                                                                      **
**  This program is an automated 1D NMR spectral simulator using                       **
**  shaped Gaussian pulses.  It runs inteactively, asking the user to                  **
**  supply a parameter file filename as well as plot parameters.  The                  **
**  system input is simply pulsed by the specified Gaussian.  It is a                  **
**  modification from Gpulse0.cc in that it correctly uses the input                   **
**  pulse phase and uses the input pulse channel for the pulse/acquire                 **
**  selectivity.                                                                        **
**                                                                                      **
**  This program does not include relaxation effects.  Also, the plot                  **
**  is immediate in gnuplot, so this will die if gnuplot is that                       **
**  program is not accessible.                                                          **
**                                                                                      **
** Author:    S.A. Smith                                                                **
** Date:      7/8/96                                                                    **
** Update:    7/8/96                                                                    **
** Version:   3.5                                                                       **
**                                                                                      **
*****************************************************************************************/

#include <gamma.h>                                // Include GAMMA

main (int argc, char* argv[])

  {
  cout << "\n\n\t\t\tGAMMA 1D NMR Simulation Program";
  cout << "\n\t\t\tGaussian Pulses, No Relaxation\n\n";
//                        Read in System and Pulse Parameters

  int qn = 1;
  String filein;                                  // Input system file name
  query_parameter(argc, argv, qn++,               // Get system file name
      "\n\tInput Parameter File? ", filein);
  sys_dynamic sys;                                // A spin system
  sys.read(filein);                               // Read in the system
  Gpuldat Gdata = read_Gpulse(filein, sys);       // Read Gaussian pulse params
  print_Gpulse(cout, Gdata);                      // Print Gaussian pusle params
  sys.offsetShifts(Gdata.Wrf);                    // Center system at pulse
//                   Determine Isotope Detection Type, Set Variables

 String IsoD = Gdata.Iso;
 gen_op sigma = sigma_eq(sys);                    // Set density matrix equilibrium
 gen_op H = Ho(sys);                              // Isotropic Hamiltonian
 gen_op detect = Fm(sys, IsoD);                   // Set detection operator to F-
 acquire1D ACQ(detect, H, 1.e-3);                 // No relaxation during acquisition (Ho)
 gen_op FXY = Fxy(sys, IsoD, Gdata.phi);          // For Gaussian RF Hamiltonian
 gen_op UGauss = Gpulse_U(H,FXY,Gdata);           // Gaussian pulse propagator
//                        Apply The Pulse Sequence

 sigma = evolve(sigma, UGauss);                   // Apply Gaussian pulse
 matrix mx = ACQ.table(sigma);                    // Set up 1D acquisition
//                       Set The Plotting Parameters

 double Fstart, Fend;                             // Number of points in FID
 query_parameter(argc, argv,                      // Get number of points
   qn++, "\n\tPlot Starting Frequency? ", Fstart);
 query_parameter(argc, argv,                      // Get number of points
   qn++, "\n\tPlot Final Frequency? ", Fend);
 int N;
 query_parameter(argc, argv,                      // Get number of points
   qn++, "\n\tPlot Points? ", N);
 double lwhh;                                     // Half-height linewidth
 query_parameter(argc, argv,                      // Get number of points
   qn++, "\n\tPlot Linewidths? ", lwhh);
 offset(mx, Gdata.Wrf, lwhh, 1);                  // Set input w offset, lwhh
//                        Construct Plot and Draw

 row_vector data=ACQ.F(mx,N,Fstart,Fend,1.e-3);// Frequency acquisition
 GP_1D("Gauss.gnu", data, 0 , Fstart, Fend);      // Output in gnuplot
 GP_1Dplot("gnu.dat", "Gauss.gnu");               // Interactive 1D gnuplot
 cout << "\n";
 }
```

Gpulcorr2.cc

```
/* Gpulcorr3.cc *************************************************************************
**                                                                                     **
GAMMA Gaussian Pulse With Profiling Simulation                                         **
**                                                                                     **
** This program is an updated version of that published in the original                **
** GAMMA article, JMR 106A, 75-105 (1994).  Given an input spin system,                **
** a specified Gaussian pulse, and a few other parameters, this will                   **
** calculate an NMR spectrum following the application of the pusle.                   **
** At the same time, the Gaussian pulse exitation profile based on                     **
** response to a single spin 1/2 particle is computed.  This profile is                **
** used to phase correct the spectrum from the input spin system.  The                 **
** phase corrected spectrum and the magnitude single spin profile are                  **
** plotted to the screen using gnuplot.                                                **
**                                                                                     **
** This program is similar to Gpulcorr1.cc but it just outputs a single                **
** plot which contains the corrected spectrum and the single spin's                    **
** profile (magnitude mode).                                                           **
**                                                                                     **
** Author:    S.A. Smith                                                               **
** Date:      7/3/96                                                                   **
** Last Date: 7/8/96                                                                   **
** Copyright: S.A. Smith, July 1996                                                    **
** Limits:    1.) Needs >= GAMMA 3.5                                                   **
**            2.) Uses Gnuplot Interactively                                           **
**            3.) No relaxation effects are included.                                  **
**                                                                                     **
*****************************************************************************************/

#include <gamma.h>                                      // Include GAMMA itself

main (int argc, char** argv)

 {
 cout << "\n\n\t\t\tGAMMA 1D NMR Simulation Program";
 cout << "\n\t\t     Gaussian Pulse Profile, No Relaxation\n\n";
//                      Read in System and Pulse Parameters
 int qn = 1;                                            // Query number
 String filein;                                         // Input system file name
 query_parameter(argc, argv, qn++,                      // Get system file name
       "\n\tInput Parameter File? ", filein);
 spin_system sys;                                       // A spin system
 sys.read(filein);                                      // Read in the system
 Gpuldat Gdata = read_Gpulse(filein, sys);              // Read Gaussian pulse params
 print_Gpulse(cout, Gdata);                             // Print Gaussian pusle params
 sys.offsetShifts(Gdata.Wrf);                           // Center system at pulse
//                 Determine Isotope Detection Type, Set Variables
 String IsoD = Gdata.Iso;                               // Set detection=pulse isotope
 gen_op sigma = sigma_eq(sys);                          // Set density op at equilibrium
 gen_op H = Ho(sys);                                    // Isotropic Hamiltonian
 gen_op detect = Fm(sys, IsoD);                         // Set detection operator to F-
 acquire1D ACQ(detect, H, 1.e-3);                       // Set for acquisition, No rel.
 gen_op FXY = Fxy(sys, IsoD, Gdata.phi);                // For Gaussian RF Hamiltonian
 gen_op UGauss = Gpulse_U(H,FXY,Gdata);                 // Gaussian pulse propagator
//                           Apply The Pulse Sequence
 sigma = evolve(sigma, UGauss);                         // Apply Gaussian pulse
 matrix mx = ACQ.table(sigma);                          // Perform 1D acquisition
//                           Set Plotting Parameters
 double Fst, Fend;                                      // Number of points in FID
 query_parameter(argc, argv, qn++,                      // Get number of points
    "\n\tPlot Starting Frequency? ", Fst);
 query_parameter(argc, argv,                            // Get number of points
  qn++, "\n\tPlot Final Frequency? ", Fend);
 int N;
 query_parameter(argc, argv,                            // Get number of points
            qn++, "\n\tPlot Points? ", N);
 double lwhh;                                           // Half-height linewidth
 query_parameter(argc, argv,                            // Get number of points
      qn++, "\n\tPlot Linewidths? ", lwhh);
 offset(mx, Gdata.Wrf, lwhh, 1);                        // Set input w offset, lwhh
//                       Calculate The Single Spin Profile
//                     (Generally Determine Phase Behavior)
 spin_system sys1(1);                                   // Single spin system
 detect = Fm(sys1, IsoD);                               // Set detection operator to F-
 FXY = Fxy(sys1, IsoD, Gdata.phi);                      // For Gaussian RF Hamiltonian
 gen_op sigma0 = sigma_eq(sys1);                        // Initial density matrix
 row_vector profile(N);                                 // Block for the profile
 double voff = Fst;                                     // Begin @ 1st plotted frequency
 double delv = (Fend-Fst)/double(N-1);                  // Offset increment
 complex z;
 for(int offs=0; offs<N; offs++)                        // Loop offsets
  {
  sys1.shift(0, voff);                                  // Set spin chemical shift
  sys1.offsetShifts(Gdata.Wrf);                         // In pulse rotating frame
  H = Ho(sys1);                                         // Calculate the Hamiltonian
  UGauss = Gpulse_U(H, FXY, Gdata);                     // Gaussian pulse propagator
  sigma = evolve(sigma0, UGauss);                       // Apply Gaussian pulse
  z = trace(detect, sigma);                             // Get magnetization
  profile.put(norm(z), offs);                           // Store magnitude
  voff += delv;                                         // Adjust the offset
  }
 GP_1D("profile.gnu",profile,0,Fst,Fend);               // Output in gnuplot
//      Perform the Phase Correction Using Phases for a 1-Spin System
//    Done For Each Transition in the Spin System and Corrected Discretely
```

```
 double w;
 complex zel;
 int ntr = mx.rows();                          // Number of transitions
 for(int tr=0; tr<ntr; tr++)                   // Loop the transitions
   {
   w = mx.getIm(tr,0)/(2.0*PI);                // Transition frequency (Hz)
   sys1.shift(0, w);                           // Set spin chemical shift
   sys1.offsetShifts(Gdata.Wrf);               // In pulse rotating frame
   H = Ho(sys1);                               // Calculate the Hamiltonian
   UGauss = Gpulse_U(H, FXY, Gdata);           // Gaussian pulse propagator
   sigma = evolve(sigma0, UGauss);             // Apply Gaussian pulse
   z = trace(detect, sigma);                   // Get transverse magnetization
   zel = mx.get(tr,1);                         // Large sytem transition intensity
   zel *= norm(z)/z;                           // Adjust transition phase
   mx.put(zel,tr,1);                           // Reset (adjusted) intensity
   }
 row_vector spec = ACQ.F(mx,N,Fst,Fend,1.e-3); // Frequency acquisition
 GP_1D("spec.gnu",spec,0,Fst,Fend);            // Output in gnuplot
//      Now Output the Corrected Spectrum & Profile Magnitude to Screen
 cout << "\n\n";                               // Keep screen nice
 cout.flush();                                 // Also keeps screen nice
 ofstream gnuload("gnu.dat");                  // File of gnuplot commands
 gnuload << "set data style line\n";           // Set 1D plots to use lines
 gnuload << "set xlabel \"W(Hz)\"\n";          // Set X axis label
 gnuload << "set ylabel \"Intensity\"\n";      // Set Y axis label
 gnuload << "set title\"Spectrum\"\n";         // Set plot title
 gnuload << "plot \"spec.gnu\"";               // Command to plot both
 gnuload << ", \"profile.gnu\"\n";             // at the same time
 gnuload << "pause -1 \'<Return> To Exit \n";  // Pause before exit
 gnuload << "exit\n";                          // Now exit gnuplot
 gnuload.close();                              // Close gnuplot command file
 system("gnuplot \"gnu.dat\"\n");              // Invoke gnuplot now
 cout << "\n\n";                               // Keep the screen nice
 }
```

Gpulpcorr2.cc

```
/* Gpulpcorr2.cc ***********************************************************************
**                                                                                    **
**          GAMMA Gaussian Pulse With 1st Order Phase Correction                      **
**                                                                                    **
** This program applies a Gaussian pulse to an arbitrary spin system.                 **
** The resulting spectrum is then adjusted by a common 1st order phase                **
** correction.  The idea is that we'd like to see how well the phase                  **
** corrections available in the spectrometer handle fixing the phase                  **
** problems resulting from selective Gaussian pulses.                                 **
**                                                                                    **
** Author:    S.A. Smith                                                              **
** Date:      7/9/96                                                                  **
** Last Date: 7/9/96                                                                  **
** Copyright: S.A. Smith, July 1996                                                   **
** Limits:    1.) Needs >= GAMMA 3.5                                                  **
**            2.) Uses Gnuplot Interactively                                          **
**            3.) No relaxation effects are included.                                 **
**                                                                                    **
***************************************************************************************/

#include <gamma.h>                                  // Include GAMMA itself

main (int argc, char** argv)

  {
  cout << "\n\n\t\t\tGAMMA 1D NMR Simulation Program";
  cout << "\n\t\t   Gaussian Pulse, 1st Order Phase Correction\n\n";
//                Read in System and Pulse Parameters
  int qn = 1;                                       // Query number
  String filein;                                    // Input system file name
  query_parameter(argc, argv, qn++,                 // Get system file name
             "\n\tInput Parameter File? ", filein);
  spin_system sys;                                  // A spin system
  sys.read(filein);                                 // Read in the system
  Gpuldat Gdata = read_Gpulse(filein, sys);         // Read Gaussian pulse params
  print_Gpulse(cout, Gdata);                        // Print Gaussian pusle params
  sys.offsetShifts(Gdata.Wrf);                      // Center system at pulse
//              Determine Isotope Detection Type, Set Variables
  String IsoD = Gdata.Iso;                          // Set detection=pulse isotope
  gen_op sigma = sigma_eq(sys);                     // Set density op at equilibrium
  gen_op H = Ho(sys);                               // Isotropic Hamiltonian
  gen_op detect = Fm(sys, IsoD);                    // Set detection operator to F-
  acquire1D ACQ(detect, H, 1.e-3);                  // Set for acquisition, No rel.
  gen_op FXY = Fxy(sys, IsoD, Gdata.phi);           // For Gaussian RF Hamiltonian
  gen_op UGauss = Gpulse_U(H,FXY,Gdata);            // Gaussian pulse propagator
//                     Apply The Pulse Sequence
  sigma = evolve(sigma, UGauss);                    // Apply Gaussian pulse
  matrix mx = ACQ.table(sigma);                     // Perform 1D acquisition
//                     Set Plotting Parameters
  double Fst, Fend;                                 // Number of points in FID
  query_parameter(argc, argv, qn++,                 // Get number of points
     "\n\tPlot Starting Frequency? ", Fst);
  query_parameter(argc, argv,                       // Get number of points
   qn++, "\n\tPlot Final Frequency? ", Fend);
  int N;
  query_parameter(argc, argv,                       // Get number of points
           qn++, "\n\tPlot Points? ", N);
  double lwhh;                                      // Half-height linewidth
  query_parameter(argc, argv,                       // Get number of points
       qn++, "\n\tPlot Linewidths? ", lwhh);
  offset(mx, Gdata.Wrf, lwhh, 1);                   // Set input w offset, lwhh
  row_vector spec = ACQ.F(mx,N,Fst,Fend,1.e-3);     // Frequency acquisition
  GP_1D("spec.gnu",spec,0,Fst,Fend);                // Output in gnuplot
//         Calculate The 1st Order Phase Corrected Spectrum
  pcorrect(mx, Gdata.Wrf, Fend, 10);                // Phase correct
  row_vector paspec=ACQ.F(mx,N,Fst,Fend,1.e-3);     // Frequency acquisition
  GP_1D("paspec.gnu", paspec, 0, Fst, Fend);        // Output in gnuplot
//      Now Output the Spectrum & Phase Corrected Spectrum to Screen
  cout << "\n\n";                                   // Keep screen nice
  cout.flush();                                     // Also keeps screen nice
  ofstream gnuload("gnu.dat");                      // File of gnuplot commands
  gnuload << "set data style line\n";               // Set 1D plots to use lines
  gnuload << "set xlabel \"W(Hz)\"\n";              // Set X axis label
  gnuload << "set ylabel \"Intensity\"\n";          // Set Y axis label
  gnuload << "set title\"Spectrum\"\n";             // Set plot title
  gnuload << "plot \"spec.gnu\"";                   // Command to plot both
  gnuload << ", \"paspec.gnu\"\n";                  // at the same time
  gnuload << "pause -1 \'<Return> To Exit \n";      // Pause before exit
  gnuload << "exit\n";                              // Now exit gnuplot
  gnuload.close();                                  // Close gnuplot command file
  system("gnuplot \"gnu.dat\"\n");                  // Invoke gnuplot now
  cout << "\n\n";                                   // Keep the screen nice
  }
```

Gplot.cc

Generate Plots of Gaussian Pulse Waveforms

```
/* Gplot.cc ******************************************************************-*-c++-*-
**                                                                              **
** This program plots the rf-field amplitude versus time for                    **
** given a Gaussian pulse as specified by four parameters:                      **
**                                                                              **
** 1.) The field strength at maximum                                            **
** 2.) The intensity cutoff (%) at the pulse endpoints                          **
** 3.) The number of steps to take for the pulse                                **
** 4.) The time over which the pulse is active                                  **
**                                                                              **
** A gaussian function centered about time t  is given formally by              **
**                                         o                                    **
**                                                                              **
**                    [        2 /           2  ]                               **
**         G(t) = exp | -(t-t ) / (2*sigma ) |                                  **
**                    [      0 /                ]                               **
**                                                                              **
** The discrete function is similar except we would like to define              **
** sigma in terms of a cutoff.  That is to say, we should like to set           **
** the Gaussian linewidth such that the first and last points are at            **
** a set percentage (of maximum == 1).                                          **
**                                                                              **
** For a cutoff of X%, we need to satisfy the following conditions              **
**                                                                              **
**                          2          2                                        **
**            0.0X = exp(-N / 8*sigma )                                         **
**                                                                              **
** or                                                                           **
**           sigma = N / sqrt[-8*ln(0.0X)]                                      **
**                                                                              **
** where N is the number of Gaussian steps taken and N/2 is the peak            **
** maximum.  Setting the peak maximum to be related to an rf-field              **
** strength, this leaves us with the formula                                    **
**                                                                              **
**                   [         2              2 ]                               **
**         G(i) = exp | (2i-N) ln(0.0X)/ N  |                                   **
**                   [                         ]                                **
**                                                                              **
** The output is sent directly to the screen using Gnuplot.  The user           **
** may also have a plot output in Framemaker MIF format.                        **
**                                                                              **
**                                                                              **
** Author:     S.A. Smith                                                       **
** Date:       May 2 1995                                                       **
** Last Update: May 8 1995                                                      **
**                                                                              **
******************************************************************************* **/

#include <gamma.h>                                    // Include all of GAMMA

main(int argc, char* argv[])
  {
  int qn = 1;                                         // Query number
  int npts;                                           // Number of points
  double gamB1, time, fact;                           // Gaussian pulse parameters
  ask_Gpulse(argc, argv, qn,                          // Get pulse parameters
             npts, gamB1, time, fact, 1);
  row_vector G = Gvect(gamB1,npts,fact);              // Fill vector with waveform
  cout << "\n\n";                                     // Keep screen nice
  GP_1D("Gauss.gnu", G);                              // Output rf lineshape gnuplot
  ofstream gnuload("gnu.dat");                        // File of gnuplot commands
  gnuload << "set data style line\n";                 // Set 1D plots to use lines
  gnuload << "set xlabel \"time(sec)\"\n";            // Set X axis label
  gnuload << "set ylabel \"gamB1(Hz)\"\n";            // Set Y axis label
  gnuload << "set title\"Gaussian Pulse Shape\"\n";   // Set plot title
  gnuload << "plot \"Gauss.gnu\"\n";                  // Command to plot
  gnuload << "pause -1 \'<Return> To Exit \n";        // Command to pause
  gnuload << "exit\n";                                // Command to exit gnuplot
  gnuload.close();                                    // Close gnuplot command file
  system("gnuplot \"gnu.dat\"\n");                    // Now, actually run gnuplot
  String syn;                                         // When plot is complete, see
  cout << "\n\n\tFrameMaker Hardcopy[y/n]? "          ;// if output in FrameMaker is
  cin >> syn;                                         // desired.
  if(syn == "y")
    FM_1D("Gauss.mif", G, 1);                         // Output to Framemaker
  cout << "\n\n";                                     // Keep the screen nice
  }
```

Ghistplot.cc

Histogram Plots of Gaussian Pulse Waveforms

```
/* Ghistplot.cc ***************************************************************-*-c++-*-
**                                                                          **
**  This program plots the rf-field amplitude versus time for a             **
**  given a Gaussian pulse as specified by four parameters:                 **
**                                                                          **
**  1.) The field strength at maximum                                       **
**  2.) The intensity cutoff (%) at the pulse endpoints                     **
**  3.) The number of steps to take for the pulse                           **
**  4.) The time over which the pulse is active                             **
**                                                                          **
**  A gaussian function centered about time t  is given formally by         **
**                                           o                              **
**                                                                          **
**                  [        2 /        2  ]                                **
**          G(t) = exp | -(t-t ) / (2*sigma ) |                             **
**                  [      0 /             ]                                **
**                                                                          **
**  The discrete function is similar except we would like to define        **
**  sigma in terms of a cutoff.  That is to say, we should like to set      **
**  the Gaussian linewidth such that the first and last points are at       **
**  a set percentage (of maximum == 1).                                     **
**                                                                          **
**  For a cutoff of X%, we need to satisfy the following conditions         **
**                                                                          **
**                          2                                               **
2                                                                           **
**             0.0X = exp(-N / 8*sigma )                                    **
**                                                                          **
**  or                                                                      **
**            sigma = N / sqrt[-8*ln(0.0X)]                                 **
**                                                                          **
**  where N is the number of Gaussian steps taken and N/2 is the peak       **
**  maximum.  Setting the peak maximum to be related to an rf-field         **
**  strength, this leaves us with the formula                               **
**                                                                          **
**                  [       2          2 ]                                  **
**          G(i) = exp | (2i-N) ln(0.0X)/ N  |                              **
**                  [                     ]                                 **
**                                                                          **
**  Author:      S.A. Smith                                                 **
**  Date:        May 2 1995                                                 **
**  Last Update: May 2 1995                                                 **
**                                                                          **
*****************************************************************************/

#include <gamma.h>                              // Include all of GAMMA

row_vector Gshape(double gamB1, double tau, int N, double fact=0.05)

   // Input        gamB  : The rf-field strength (Hz)
   //               tau  : Gaussian pulse length (sec)
   //               N    : Number of Gaussian steps
   //               fact : Cutoff factor
   // Output       angle: Gaussian pulse rotation angle
   //                         on resonance

 {
 if(fact>1.0 || fact<0.000001)                  // Make sure fact is between
   fact = 0.000001;                             // [0,1]
 double tdiv = tau/double(N);                   // Incremental time
 double den = double((N-1)*(N-1));              // Denominator
 double logf = log(fact);                       // Log of the cutoff factor
 double Z = logf/den;                           // Exponential factor
 double Gnorm;                                  // Normalized Gaussian intensity
 double num;
 int M = N+1;
 if(N > 100) M=0;
 row_vector Gshape(2*N+M);                      // Vector of Gaussian points
 double lastv, lastt;
 int I = 0;

 double Gs[N];
 for(int i=0; i<N; i++)                         // Loop over Gaussian steps
   {
   if(N-1-i < i)                                //Use symmetry to avoid
     Gs[i] = Gs[N-1-i];                         //recalculating same pts
   else
     {
     num = double(2*i)-double(N-1);             // Index so Gaussian mid-pulse
     Gnorm = exp(Z*num*num);                    // Normalize Gauss. amplitude
     Gs[i] = gamB1*Gnorm;                       // RF amplutude modulation
     }
   }

 double time = 0.0;                             // Time in pulse
 for(i=0; i<N; i++)                             // Loop over Gaussian steps
   {
   if(i)
     {
     Gshape.put(complex(time,Gs[i-1]),I++);     // For horizontals in hist.
     if(M) Gshape.put(complex(time,0),I++);     // For verticals in hist.
     }
   else if(M)
     Gshape.put(complex(time,0),I++);           // First vertical in hist.
   Gshape.put(complex(time,Gs[i]),I++);         // Gaussian intensity
   lastv = Gs[i];                               // Store the previous intensity
   lastt = double(i);                           // Store the previous point
```

```
   if(i==N-1)                                      // For the last point
     {
     time += tdiv;
     Gshape.put(complex(time,Gs[i]),I++);          // For last horizontal in hist.
     if(M) Gshape.put(complex(time,0),I++);        // For last vertical in hist.
     }
   time += tdiv;
   }                                               // (evolve/acq step goes here)
  return Gshape;
  }


main(int argc, char* argv[])
 {
 int qn = 1;                                       // Query number
 int npts;                                         // Number of points
 query_parameter(argc, argv, qn++,                 // Get number of steps(pts)
  "\n\tNumber of Points in Gaussian? ", npts);
 if(npts < 2) npts = 2048;
 double gamB1;
 query_parameter(argc, argv, qn++,                 // Get rf-field strength
  "\n\tRF-Field Stength (Hz)? ", gamB1);
 double time;
 query_parameter(argc, argv, qn++,                 // Get pulse length
  "\n\tGaussian Pulse Length (sec)? ", time);
 double fact;
 query_parameter(argc, argv, qn++,                 // Get system file name
  "\n\tPercent Intensity at Ends [0, 1]? ", fact);
 cout << "\n\n";                                   // Keep the screen nice
 row_vector G = Gshape(gamB1,time,npts,fact);
 GP_xy("Gauss.gnu", G);                            // Output rf lineshape gnuplot
 ofstream gnuload("gnu.dat");                      // File of gnuplot commands
 gnuload << "set data style line\n";               // Set 1D plots to use lines
 gnuload << "plot \"Gauss.gnu\"\n";                // Plot IxA in gnuplot
 gnuload << "pause -1 \'<Return> To Exit \n";      // Plot IxA in gnuplot
 gnuload << "exit\n";                              // Plot IxA in gnuplot
 gnuload.close();                                  // Close gnuplot command file
 system("gnuplot \"gnu.dat\"\n");                  // Plot to screen
 cout << "\n\n";                                   // Keep the screen nice
 }
```